On the Computer Aided Design of a Shaft Subjected to Reversed Bending and Steady Torsion

A thesis presented to
The Faculty of the College of Engineering and Technology
Ohio University

In partial fulfillment
of the requirements for the Degree of
Master of Science

by
Peter L. Curcio
June 1987
ACKNOWLEDGEMENTS

The author wishes to express his sincere thanks to Professor Dr. K.R. Halliday, whose valuable help and guidance made this project possible. The author also wishes to thank Professor O.E. Adams and Professor Ralph Rogers for their suggestions and encouragement.
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Previous Work</td>
<td>2</td>
</tr>
<tr>
<td>1.3</td>
<td>Goal of Work</td>
<td>4</td>
</tr>
<tr>
<td>1.4</td>
<td>Program Description</td>
<td>4</td>
</tr>
<tr>
<td>2.1</td>
<td>Introduction</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>Model of the Shaft</td>
<td>16</td>
</tr>
<tr>
<td>2.3</td>
<td>Support Conditions</td>
<td>19</td>
</tr>
<tr>
<td>2.4</td>
<td>Design Approach</td>
<td>23</td>
</tr>
<tr>
<td>2.5</td>
<td>Static and Fatigue Analysis</td>
<td>24</td>
</tr>
<tr>
<td>2.6</td>
<td>Design Process</td>
<td>25</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>27</td>
</tr>
<tr>
<td>3.2</td>
<td>Sample Run</td>
<td>27</td>
</tr>
<tr>
<td>3.3</td>
<td>Limitations and Future Enhancements</td>
<td>42</td>
</tr>
</tbody>
</table>

BIBLIOGRAPHY

Appendix A
Flowchart
Appendix B
Stress Concentration Curve Fits and Data
Appendix C
Source Code Listing

Page 45
<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>Continuous Model Replaced by Discrete Points</td>
<td>17</td>
</tr>
<tr>
<td>2-2</td>
<td>Fixed End Support</td>
<td>19</td>
</tr>
<tr>
<td>2-3</td>
<td>Free End and Intermediate Support</td>
<td>21</td>
</tr>
<tr>
<td>2-4</td>
<td>Intermediate Support</td>
<td>22</td>
</tr>
<tr>
<td>3-1</td>
<td>Question Screen 1</td>
<td>28</td>
</tr>
<tr>
<td>3-2</td>
<td>Question Screen 2</td>
<td>28</td>
</tr>
<tr>
<td>3-3</td>
<td>Design Area</td>
<td>29</td>
</tr>
<tr>
<td>3-4</td>
<td>Question Screen for the Hub and Load Information</td>
<td>30</td>
</tr>
<tr>
<td>3-5</td>
<td>Diameter Adjustment Messages</td>
<td>31</td>
</tr>
<tr>
<td>3-6</td>
<td>Shear Diagram</td>
<td>32</td>
</tr>
<tr>
<td>3-7</td>
<td>Moment Diagram</td>
<td>33</td>
</tr>
<tr>
<td>3-8</td>
<td>Bearing Selection Screen</td>
<td>34</td>
</tr>
<tr>
<td>3-9</td>
<td>Location of Grooves, Keyways, and Holes</td>
<td>35</td>
</tr>
<tr>
<td>3-10</td>
<td>Preliminary Design Layout</td>
<td>36</td>
</tr>
<tr>
<td>3-11</td>
<td>Graph of M/EI</td>
<td>37</td>
</tr>
<tr>
<td>3-12</td>
<td>Effects of Stress Concentration</td>
<td>38</td>
</tr>
<tr>
<td>3-13</td>
<td>Endurance Limit Modifying Factors</td>
<td>39</td>
</tr>
<tr>
<td>3-14a</td>
<td>Final Preliminary Design</td>
<td>40</td>
</tr>
<tr>
<td>3-14b</td>
<td>Final Design Dimensions</td>
<td>40</td>
</tr>
<tr>
<td>3-15</td>
<td>Notify the User Design Criteria not met</td>
<td>41</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Catalog Management Facilities</td>
<td>5</td>
</tr>
<tr>
<td>1-2</td>
<td>Bearing Specifications</td>
<td>6</td>
</tr>
<tr>
<td>1-3</td>
<td>Typical Range of Surface Roughness</td>
<td>13</td>
</tr>
</tbody>
</table>
NOMENCLATURE

L = length of the shaft
Se = fully corrected endurance limit
S'e = endurance limit of rotating-beam specimen
Ka = surface factor
Kb = size factor
Kc = reliability factor
Kd = temperature factor
Ke = stress concentration factor
Kf = miscellaneous-effects factor
d4y = fourth derivative of deflection
d3y = third derivative of deflection
d2y = second derivative of deflection
h = distance between discrete points, inches
E = modulus of elasticity, psi
I = moment of inertia, in4
q(x) = load per unit length
n = factor of safety
π = pi
T = torque, inch-pounds
M = bending moment, inch-pounds
Sy = yield strength, psi
Sult = ultimate tensile strength, psi
CHAPTER I

1.1 Introduction

The personal computer (PC) is becoming an invaluable tool for both the engineer and the scientist. This is due to the growing wealth of hardware and software currently available for the PC. Most of the scientific and engineering computing in the 1970's was performed on a mainframe computer. Today, the computing power of a mainframe computer can be handled by innovations in personal computer hardware and software such as the hard disk, expandible memory, high resolution monitors, data acquisition equipment, wordprocessing software, finite element programs, computer drafting and interactive graphics. All of these have helped to revolutionize the usefulness of the PC. These facilities are currently available as problem solving aids which the engineer has at his or her disposal. The purpose of this thesis is to develop an IBM Personal Computer based Computer Aided Design (CAD) program for the preliminary design of a shaft. A more appropriate label which may be devised is Computer Aided Engineering (CAE) since the computer acts only as an aid in the design process.
All of the previous work sheds some light on aspects of a complex problem. There is a need for software which has more built in flexibility to account for most all the possible design configurations and has the capability to graphically represent the results as they are generated. A system with the ability to account for most of the problem variables is an expert system.

Weiss and Kulikowski (1984) define an expert system as one which can handle real world, complex problems requiring an expert's interpretation. An expert system solves these problems using a computer model of expert human reasoning and reaches the same conclusions that a human expert would reach faced with a comparable problem. An expert's skill usually comes from extensive experience, and detailed specialized knowledge of the problems they handle. The experiential knowledge of the expert is hardly, if ever, found in textbooks. Without the human expert, the computer can not arrive at some of the alternative methods of solution. The expert system does not replace the design engineer, it requires the help of the engineer and assists the engineer by performing calculations. K.R. Halliday (1985) discusses the usefulness and effectiveness such an expert system can have in assessing the various alternate solutions for a given design problem.
1.3 Goal of Work

The software developed in this thesis is intended as a computer aided engineering aid in the preliminary design of a rotating shaft subjected to reverse bending and steady torsion. This loading is a very common situation and probably occurs more often than any other. The purpose of this thesis is to develop an IBM Personal Computer based Computer Aided Design (CAD) program for the design of a shaft. As information about the design is computed the results are reported to the user, usually in the form of graphical output. The designer can utilize this information to make any changes in the design or to reevaluate the entire design.

1.4 Program Description

The CAD package consists of two main programs both written in Turbo Pascal. The first is a bearing catalog data management program. This software is an expanded version of a preliminary program presented by Halliday (1985). The second program is the CAD software for the design of a shaft.

The data management program is driven interactively through a main menu. The main menu provides the user with seven facilities for creating and maintaining catalogs of different types of rolling element bearings (see table 1-1).
Table 1-1
Catalog Management Facilities
Main Menu
----------------------------------------
1. Create Catalog......C
2. Add To Catalog......A
3. Review Catalog......R
4. Print Catalog......P
5. Sort Catalog.......S
6. Edit Catalog.......E
7. Quit Program.......Q

The user selects the letter of the option to invoke one of the facilities. Upon selecting a facility the user will be prompted to enter a filename, the filename extension '.DAT' is appended automatically. The following discussion describes each of these facilities.

To create a file the user selects the create facility and enters the filename. The program then echoes the filename and warns the user that any file previously declared under this name will be destroyed. The user may or may not continue by entering 'Y' or 'N'. Having named a catalog the user is prompted to choose either English or Metric units and will be prompted for the bearing specifications (see table 1-2).
Table 1-2
Bearing Specifications

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Part Number</th>
<th>Bore Diameter</th>
<th>Outside Diameter</th>
<th>Width</th>
<th>Fillet Radius</th>
<th>Shoulder Diameter of Inner Race</th>
<th>Shoulder Diameter of Outer Race</th>
<th>Load Rating</th>
</tr>
</thead>
</table>

The program supplies the user with five choices of bearing vendors. Choices one through four are common bearing manufacturers, the fifth is an option which can be used to satisfy a special requirement. The remaining specs, with the exception of the load rating, are geometric characteristics of the bearing. The load rating is the manufacturer's specified dynamic load rating. The order in which the entries are placed in the file is not important providing the user uses the sort facility. Sorting places the bearings in order of increasing load rating. The catalog has to be sorted so that the shaft design program can properly select the bearings to be seated on the shaft.

To edit a file the user selects the edit facility and enters the catalog filename. The program writes to the screen the catalog and the number of the entry the user is editing. The user is then given three choices; delete, edit or quit the facility. Delete will erase the current entry from the catalog. Edit will allow the user to change any of the bearing specifications. Quitting will exit the
editor, update the bearing catalog and return the user to the main menu.

The remaining facilities work in much the same way and will be discussed briefly. Adding to the catalog allows the user to put additional bearing entries into an existing catalog. Again, after entering the new entries the user must use the sort facility. Reviewing the catalog outputs the catalog entries to the screen. Printing enables the user to get a hardcopy of a catalog. Sorting places the catalog entries in order of increasing load rating. Quitting exits the program and returns the user to DOS or the system environment.

The second major software package developed during this project is the shaft design program. The shaft design program prompts the user with a series of question screens which are designed to provide information needed to design the shaft. The first screen asks the user to specify either carbon steel or aluminum as the choice of material and to supply the yield strength and the ultimate tensile strength. The second screen asks for the shaft length, the number of bearings and their location from the left end of the shaft. All prompts asking for the location of items on the shaft are referenced from the left end. The design area is now drawn on the screen showing the shaft centerline, bearing locations and a reference set of coordinate axes. When the graphical output is complete and
all questions have been answered, the user must press the space bar to resume normal execution. The user is not prompted to press the space bar and must remember to on their own. The third question screen asks for the number of hubs, width, location, concentrated load, torque and axial thrust which is to be assigned to that hub. The sense of the loads follows standard beam convention. A positive load acts down, a positive torque follows the right hand rule and a positive axial thrust acts toward the right along the shaft axis.

After the loading on the hubs has been inputted, the program calculates the shaft deflection. From the length of the shaft a minimum starting diameter of $L/20$ is assumed. This minimum is chosen for buckling and vibration considerations. The shaft is then tested for deflection. If the maximum deflection of the shaft is greater than 0.01 inches then the diameter is increased accordingly and the program calculates the new deflections until this constraint is met. The constraint on the deflection has to be met to insure the proper mesh or alignment of components on the shaft. The bending deflection analysis is performed by the well-known finite difference technique modeling the shaft as a beam. If the number of bearings is greater than two the problem becomes statically indeterminate and lends itself to tedious calculations. The results from the
deflection analysis are necessary to continue the analysis of the forces on the shaft.

Using the deflections the program calculates the shear forces, bending moments and the bearing reactions. Also the torque throughout the shaft is calculated and the axial thrust is found. The axial thrust is assumed to act on the bearing at the compressive end of the shaft. The program models the bearings as simple supports. A beeping signal alerts the user at the completion of the calculations. The results of these calculations are then displayed graphically. If the user has an Epson series MX, RX or FX printer hardcopies of the graphs can be obtained under program control, see pages 27 through 39 of this thesis. The first two graphs are the shear and moment diagrams. After they are displayed the user is given the opportunity to get a hardcopy of the loading and force distribution on the shaft.

The selection of bearings follows from the determination of the bearing reactions. The user is asked to supply the information necessary to determine a suitable bearing. The bearings are selected according to the largest calculated load rating based on the procedure outlined by Shigley (1983). The bearing catalog is searched until a bearing is found with an appropriate diameter and load rating. If a suitable bearing can not be
found the user is notified and the largest bearing in the catalog is selected in order to continue the design.

To secure the components placed on the shaft grooves, holes and keyways need to be incorporated. The user is prompted for the tentative location of each of these. A layout is then drawn with the bearings, hubs, grooves, holes and keyways located on the shaft. The sections of the shaft are labeled so the user can select the region whose diameter must be stepped in order to properly locate, or seat, the bearings and hubs. After selecting a shaft section, the user will be prompted to supply the fillet radii. The step diameters are then computed and the user is given the opportunity to change the information on the grooves, holes and keyways. If there are no changes the user is asked to supply the respective information on these geometric factors, for example, the diameter of the hole or the radius of the groove. The existence of notches or discontinuities alter the stresses in the shaft and must be accounted for. The effects of notch sensitivity are taken into account by empirical equations which have been fitted to the stress concentration graphs published by R.E. Peterson (1953).

Given the loading and new geometry the shaft design has to be checked against static and fatigue failure. When mechanical parts are subjected to a large number of alternating or fluctuating stress cycles, which is
frequently the case, the parts can fail due to fatigue. The approach taken in the fatigue analysis of this problem is the one developed by C.R. Soderberg (1935). There are two areas of the shaft to be considered, the first is the area of maximum bending effect determined from the distribution of the quantity $M/EI$ and the second is the area of maximum torsional effect determined from the local value of the torsional shear $16T/\pi D^3$.

Before testing for fatigue, the fully corrected endurance limit, denoted $S_e$, of the material must be determined. The fully corrected endurance limit, $S_e$, is equal to the product of the endurance limit modifying factors and the endurance limit, $S'e$, of a rotating beam specimen. This method will be presented after the following discussion. The rotating-beam specimen is a carefully machined and polished sample of the same type of material used in the manufacture of the mechanical part. This specimen is tested under the action of fatigue loads while the stress reversals are counted until failure. The most widely used fatigue test is the R.R. Moore high-speed rotating-beam machine. The only loading this machine subjects the specimen to is pure bending. The endurance limit, $S'e$, of the rotating beam specimen is a measure of the fatigue strength of a material at no less than one million reversed load cycles. The fully corrected endurance limit, $S_e$, is the endurance limit modified to account for
factors influencing the material strength. Since the rotating-beam test is used in the laboratory under carefully controlled conditions, it is unrealistic to expect that a mechanical part in service perform as well. To account for these conditions, factors are applied to modify the endurance limit of the beam specimen. This method described by Shigley (1983) and is presented here for completeness,

\[ Se = Ka \times Kb \times Kc \times Kd \times Ke \times Kf \times S'e \]  

(1-1)

where

- \( Se \) = fully corrected endurance limit of mechanical part
- \( S'e \) = endurance limit of rotating-beam specimen
- \( Ka \) = surface factor
- \( Kb \) = size factor
- \( Kc \) = reliability factor
- \( Kd \) = temperature factor
- \( Ke \) = stress concentration factor
- \( Kf \) = miscellaneous-effects factor

The last question screen, shown on page 39, prompts the user for information on the endurance-limit modifying factors. Under most operating conditions \( Kd \) and \( Kf \) are equal to one. If \( Kd \) and \( Kf \) equal one then the shaft; is operating at a temperature below 350 degrees Celsius, is not operating in a corrosive environment and has no metal plating. The first modifying factor, the surface roughness factor, can be chosen on the basis of a typical range of roughness given in micro-inches, as shown in table 1-3. (Source: Machinery's Handbook, 20th ed., Industrial Press, 1975, p.2395)
Table 1-3
Typical Range of Surface Roughness

<table>
<thead>
<tr>
<th>Process</th>
<th>Roughness µin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shaping</td>
<td>500-63</td>
</tr>
<tr>
<td>Drilling</td>
<td>250-63</td>
</tr>
<tr>
<td>Milling</td>
<td>250-32</td>
</tr>
<tr>
<td>Broaching</td>
<td>125-32</td>
</tr>
<tr>
<td>Reaming</td>
<td>125-32</td>
</tr>
<tr>
<td>Lathe Work</td>
<td>250-16</td>
</tr>
<tr>
<td>Grinding</td>
<td>63-4</td>
</tr>
<tr>
<td>Polishing</td>
<td>16-4</td>
</tr>
<tr>
<td>Lapping</td>
<td>16-2</td>
</tr>
<tr>
<td>Sand Casting</td>
<td>1000-500</td>
</tr>
<tr>
<td>Investment Casting</td>
<td>125-63</td>
</tr>
<tr>
<td>Extruding</td>
<td>125-32</td>
</tr>
<tr>
<td>Cold Drawing</td>
<td>125-32</td>
</tr>
<tr>
<td>Die Casting</td>
<td>63-32</td>
</tr>
</tbody>
</table>

K.R. Halliday (1985) has performed a double least squares quadratic regression on data yielding an expression for the surface factor as a function of the ultimate tensile strength and surface roughness in microinches. The size factor, $K_b$, is chosen automatically from the diameter of the shaft where the worst bending effect occurs. Options for the desired reliability factor, $K_c$, are listed so the user may choose appropriately. The determination of a strength reduction factor, $K_e$, is based on the worst effects of stress concentration or notch sensitivity.

A factor of safety, $n$, for the design is supplied by the user and the program checks for yielding under the static loading. The program informs the user whether or not the shaft has met this criteria and if not the program provides the necessary diameter and the diameter where yielding occurred. The check against fatigue is carried
out and again the user is informed as before. At this point, if the shaft has met both the static and fatigue criteria the final configuration of the shaft, with all the associated components, is displayed and the diameters and section lengths of the shaft are output to the screen. If one of the criteria is not met the process is repeated but the user has the option of changing or retaining aspects of the original configuration.

Some of the more important assumptions which were made in this analysis are:
1. infinite life of the shaft
2. deflection constraint of 0.01 inches
3. all bending stresses are alternating
4. axial and torsional stresses are mean
5. the material is homogenous.

The assumptions made correlate with the most common circumstances found in practice, with the exception of the deflection constraint which depends on the user's application. A maximum deflection of 0.01 inches is generally acceptable for most cases where the alignment of components and their meshing is concerned.

The following chapters discuss the application of the finite difference method, the shaft design process and a trial run of the software. The Appendices at the end contain a flowchart on the software, raw data fits to the
stress concentration curves and a listing of the source code.
CHAPTER II

2.1 Introduction

Due to the diversity of human needs engineers find themselves faced with a variety of problems requiring a solution. The plan or procedure used in formulating a solution is design (Shigley, 1983). If the designer can create a plan of action which satisfies the human need under the given or prevailing circumstances, then the design is one of many possible solutions. These circumstances are constraints and every design is subject to them. There is no unique solution to a design and one used in practice today may not be in use tomorrow. In terms of the mechanical engineer, mechanical design is the designing of systems, devices, instruments, products and machines (Shigley, 1983). There are many phases to design which can be categorized as follows: recognition of need, problem definition, analysis and evaluation. To analyze the design of the system, device or element a model is devised. The model is usually mathematically formulated to simulate the real problem.

2.2 Model of the Shaft

The shaft is closely analogous to a continuous beam of circular cross section with intermediate supports and point loads. Assuming the designer will provide an axial constraint, a one dimensional model can be used. This
model should be flexible enough to handle any point loading and support configuration. A general means of formulation can be accomplished using the finite difference method (or central difference method).

Stephen H. Crandall (1956) outlines a procedure for the replacement of a continuous domain by a pattern of discrete points within the domain as shown in figure 2.1. Mathematically, the governing differential equation describing the model is replaced by finite difference approximations at each discrete point. This leads to a system of n simultaneous algebraic equations, where n is the number of discrete points in the domain. In the case of the shaft model these equations are linear. The methods of matrix math can be applied to solve the system of equations.

![Fig. 2-1. Continuous Model Replaced by Discrete Points](image)

The quantities necessary to perform an analysis of the shaft using finite difference are the deflection, shear and bending moment. The axial thrust and torque are handled apart from the finite difference by summing the forces in their respective directions. Since the loading on the
shaft will be supplied by the user, a finite difference expression for the loading in terms of the deflection at each point is given by the following equations:

\[
\frac{d^4y}{dx^4} = \frac{1}{h^4} \left[ Y_{j-2} - 4Y_{j-1} + 6Y_j - 4Y_{j+1} + Y_{j+2} \right] \quad (2-1)
\]

\[
\frac{d^4y}{dx^4} = \frac{q(x)}{EI} \quad (2-2)
\]

\[
q(x) = \frac{E}{h^4} \left[ Y_{j-2} - 4Y_{j-1} + 6Y_j - 4Y_{j+1} + Y_{j+2} \right] \quad (2-3)
\]

where;
- \(d^4y\) = fourth derivative of the deflection
- \(h\) = distance between discrete points
- \(Y_j\) = deflection of the \(j\)th discrete point
- \(E\) = modulus of elasticity
- \(I\) = moment of inertia
- \(q(x)\) = load per unit length

Writing equation 2-3 at every discrete point yields a system of linear simultaneous equations of the form shown in 2-4.

\[
\begin{align*}
    a_{j-2}Y_{j-2} + a_{j-1}Y_{j-1} + a_{j}Y_{j} + a_{j+1}Y_{j+1} + a_{j+2}Y_{j+2} &= q_1 \\
    \vdots & \\
    a_{n,j-2}Y_{j-2} + a_{n,j-1}Y_{j-1} + a_{n,j}Y_{j} + a_{n,j+1}Y_{j+1} + a_{n,j+2}Y_{j+2} &= q_n
\end{align*}
\]

where;
- \(a_{j,j-2} = 1\)
- \(a_{j,j-1} = -4\)
- \(a_{j,j} = 6\)
- \(a_{j,j+1} = -4\)
- \(a_{j,j+2} = 1\)
By using matrix methods, the deflections $Y_j$, can be found. Once the deflections are found the results can be applied to the finite difference equations for shear and bending moment as follows:

\[
\frac{d^2y}{dx^2} = \frac{-1y_{j-2} + 2y_{j-1} + 0y_j - 2y_{j+1} + 1y_{j+2}}{2h^3} \quad (2-5)
\]

\[
\frac{d^3y}{dx^3} = \frac{V}{EI} \quad (2-6)
\]

\[
V = EI\left[-1y_{j-2} + 2y_{j-1} + 0y_j - 2y_{j+1} + 1y_{j+2}\right] \quad (2-7)
\]

\[
\frac{d^2y}{dx^2} = \frac{[y_{j-1} - 2y_j + y_{j+1}]}{h^2} \quad (2-8)
\]

\[
\frac{d^2y}{dx^2} = \frac{M}{EI} \quad (2-9)
\]

\[
M = EI\left[y_{j-1} - 2y_j + y_{j+1}\right] \quad (2-10)
\]

2.3 Support Conditions

The method of applying finite difference to the model of the shaft must account for various supporting conditions. The support conditions that can exist are fixed end(s), between end(s) or both. If the end(s) of the shaft are fixed, as shown in figure 2-2, the equations will start at point 2, since point 1 is fixed and the deflection there is known.

\[
-1 \quad 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad ...
\]

\[
\text{Fig. 2-2. Fixed End Support}
\]
Writing the difference equations starting at point 2 yields:

At \( j=2 \),
\[
    y_0 - 4y_1 + 6y_2 - 4y_3 + y_4 = \frac{q(x)h^4}{EI} \quad (2-11)
\]

Given \( Y_1 = 0 \),
\[
    y_0 + 6y_2 - 4y_3 + y_4 = \frac{q(x)h^4}{EI} \quad (2-12)
\]

Inspecting this equation, the deflection at \( Y_0 \) lies outside the boundary. An expression for \( Y_0 \) in terms of the point(s) within the boundary can be derived from the boundary conditions that the bending moment at point 1 equals zero. Writing the difference equation for the moment at point 1 and setting it equal to zero yields:

At \( j=1 \),
\[
    M = 0 = y_0 - 2y_1 + y_2 \quad (2-13)
\]

Given \( Y_1 = 0 \),
\[
    y_0 = -y_2 \quad (2-14)
\]

Substituting 2-14 into 2-11 and simplifying gives:
\[
    5y_2 - 4y_3 + y_4 = \frac{q(x)h^4}{EI} \quad (2-15)
\]

At \( j=3 \),
\[
    -4y_2 + 6y_3 - 4y_4 + y_5 = \frac{q(x)h^4}{EI} \quad (2-16)
\]

At \( j=4 \),
\[
    y_2 - 4y_3 + 6y_4 - 4y_5 + y_6 = \frac{q(x)h^4}{EI} \quad (2-17)
\]

From point 4 the equations resume the standard difference expressions.

If the end of the shaft is free, as depicted in figure 2-3, the points lying outside the boundary need to be expressed in terms of the points within the boundary. At
the end of shaft the shear and bending moment are equal to zero.

\[ -1 0 1 2 3 4 5 6 \ldots \]

\[ j^{-1} j j^+1 j^+2 \]

**Fig. 2-3.** Free End and Intermediate Support

Thus, the expressions for \( Y_1 \) and \( Y_0 \) are:

At \( j=1 \),

\[ M = 0 = y_0 - 2y_1 + y_2 \]  
(2-18)

\[ \therefore y_0 = 2y_1 - y_2 \]  
(2-19)

At \( j=1 \),

\[ V = 0 = -y_{-1} + 2y_0 + 0y_1 - 2y_2 + y_3 \]  
(2-20)

\[ y_{-1} = 2y_0 - 2y_2 + y_3 \]  
(2-21)

Substituting 2-19 into 2-21 and simplifying:

\[ \therefore y_{-1} = 4y_1 - 4y_2 + y_3 \]  
(2-22)

Starting at point 1 and substituting the previous expressions in the difference equations gives:

At \( j=1 \),

\[ y_{-1} - 4y_0 + 6y_1 - 4y_2 + y_3 = \frac{q(x)h^4}{EI} \]  
(2-23)

\[ 4y_1 - 4y_2 + y_3 - 4(2y_1 - y_2)y_1 - 4y_2 + y_3 = \frac{q(x)h^4}{EI} \]  
(2-24)
Simplifying 2-24, \[ 2y_1 - 4y_2 + 2y_3 = \frac{q(x)h^4}{EI} \] (2-25)

At \( j=2 \), \[ 2y_1 - y_2 - 4y_1 + 6y_2 - 4y_3 + y_4 = \frac{q(x)h^4}{EI} \] (2-26)

Simplifying 2-26 \[ -2y_1 + 5y_2 - 4y_3 + y_4 = \frac{q(x)h^4}{EI} \] (2-27)

At \( j=3 \), \[ y_1 - 4y_2 + 6y_3 - 4y_4 + y_5 = \frac{q(x)h^4}{EI} \] (2-28)

From point 3 the equations resume the standard equation form.

Where the shaft is supported at any location between the ends, as shown in figure 2-4, the finite difference expressions must be adjusted.

---

**Fig. 2-4. Intermediate Support**

To demonstrate the way in which a support condition changes the difference expressions, the equations starting at point \( j-3 \) through the support at \( j \) are:
Point

\[ \begin{align*}
  j^{-3} & : y_{j-5} + 4y_{j-4} + 6y_{j-3} - 4y_{j-2} + 6y_{j-1} + b_{j-3} \\
  j^{-2} & : y_{j-4} + 4y_{j-3} + 6y_{j-2} - 4y_{j-1} + 0 = b_{j-2} \\
  j^{-1} & : y_{j-3} - 4y_{j-2} + 6y_{j-1} + 0 = b_{j-1} \\
  j^+1 & : 0 - 6y_{j-1} - 4y_{j-2} + 6y_{j-3} + b_{j+1} \\
  j^+2 & : 0 - 4y_{j-1} + 6y_{j-2} - 4y_{j-3} + 6y_{j-4} + b_{j+2} \\
  j^+3 & : y_{j-1} - 4y_{j-2} + 6y_{j-3} - 4y_{j-4} + 6y_{j-5} + b_{j+3}
\end{align*} \]

where,

\[ b_j = \left[ \frac{q(x)h^4}{EI} \right]_{j-3} \\

\text{Where the support at } j \text{ enters into the equations it will be treated as a known quantity and ignored. The deflection here is known to be zero, therefore, the equation at this point is dropped from the difference equations. The subsequent equations affected by point } j \text{ are adjusted accordingly.}

2.4 Design Approach

C.R. Soderberg (1935) discusses a procedure in which the dimensions of a machine part can be determined in order to accommodate combinations of alternating and mean stresses. Soderberg's approach will be used to check the dimensions of the shaft against fatigue failure. A shaft loaded by stationary bending and torsional moments is subjected to reversed bending and steady torsional stress. This loading combination is very common and probably occurs more often than any other. The Soderberg approach for
combined stresses, is well known and derived in most standard textbooks on machine design, such as Shigley (1983). The design equation for the diameter of the shaft is:

\[ d = \left( \frac{32n}{TT} \left( \frac{T}{Sy} \right)^2 + \left( \frac{M}{Se} \right)^2 \right)^{0.3333} \]  

(2-30)

where:
- \( n \) = Factor of Safety
- \( T \) = Mean Torsional Moment
- \( Sy \) = Yield Strength
- \( M \) = Alternating Bending Moment
- \( Se \) = Fully Corrected Endurance Limit

2.5 Static and Fatigue Analysis

The static and fatigue analysis follows the procedure outlined by Shigley (1983). For the analysis of both the static and fatigue case, two critical areas on the shaft must be checked. The first is where the maximum bending moment and, if one is present, the associated torsional moment occurs. The second is where the maximum torsional moment and the associated bending moment occurs. For the static test the diameter at these areas of the shaft must be at least the diameter given by the following equation:

\[ d = \left( \frac{32n}{TT} \right) \left( \frac{M^2 + T^2}{5y} \right)^{0.3333} \]  

(2-31)

In the case of fatigue the diameter must be at least that of the diameter given by equation 2-30. George Sines (1959) presents experimental evidence that the
bending-fatigue strength is not affected by the torsional mean stress until the torsional yield strength is exceeded by 50 percent. If this is the case then equation 2-31 reduces to:

\[ d = \left( \frac{32n+1}{TTS_e} \right)^{0.3333} \]  

(2-32)

Due to the fact that torsional endurance limits may require different modification factors than those for the bending endurance limits. For example, the stress concentration factors are not the same for both bending and torsion. One solution is to use the modifying factors as strength-reducing factors instead of stress-increasing factors. Based on R.E. Peterson's (1953) curves for stress concentration, numerous curve fits were performed to use these curves in equation form. The data and curve fitting are included in Appendix B.

2.6 Design Process

For a CAD program to emulate the way an expert would design a shaft, a method similar to the one a designer would employ has to be simulated. The following discussion considers the questions a designer asks when planning a shaft design. A flow chart of the program is included in Appendix A.

The designer starts with some preconceived notion of the shaft configuration. The designer knows roughly the materials readily available, the length of the shaft, the
spacing of the bearings and hubs and the loading. This notion gradually takes on a more solid form as the design progresses. Due to the magnitude of the loading, a minimum diameter of standard stock size is determined in order that the shaft deflection be small enough so that the elements on the shaft align and mesh properly. After finding a suitable diameter, the bending moments and the bearing reactions are determined. Bearing selection based upon the worst loading condition and the minimum diameter is conducted. Based upon the assembly procedure, the steps for seating the hubs and the bearings will be determined. After this configuration has been determined the static and fatigue analysis is performed. If the design satisfies all of the failure criteria it becomes one of several possible solutions. If the design does not satisfy the criteria the design is reevaluated.
CHAPTER III

3.1 Introduction

A trial run of the software is presented so that the user may proceed through a sample problem. This sample run may be used as a user's guide. All of the screen output, questions, prompts and graphs are included in the following sample run. All questions prompting the user for input may be answered without discretion to the keyboard character(s) upper or lower case. A discussion on the limitations and future enhancements of the software will follow the sample run. A flowchart of the software and a listing of the source code is included in Appendices A and C.

3.2 Sample Run

With the computer on and the program disk inserted into the default diskdrive, the disk operating system date and time prompts will appear on the screen. Upon responding to these prompts, the user can initiate the Bearing Catalog Manager or the Shaft Design software. The Bearing Catalog Manager is a interactive menu driven database program. A detailed discussion of this software is given on pages 4 through 7 of this thesis. Before executing the Shaft Design software, it is helpful if the user makes a sketch of the shaft and marks off the locations of the bearings and hubs. To start the design software the user types the word [SHAFT].
The program automatically begins execution and prompts the user with a series of question screens which are designed to compile information about the design. The user enters the choice of a material for the shaft and the material's associated properties. The user then inputs the length of the shaft and the location of the bearings (located from the left end of the shaft). This is depicted in the first two screens shown in figures 3-1 and 3-2.

**** MATERIAL INFORMATION ****
MATERIAL 1. CARBON STEEL 2. ALUMINUM
CHOOSE A NUMBER » 1
ENTER THE YIELD STRENGTH (psi) » 8.4e4
ENTER THE ULTIMATE TENSILE STRENGTH (psi) » 1.0e5

Fig. 3-1. Question Screen 1

**** SHAFT LENGTH & BEARING INFORMATION ****
ENTER THE LENGTH (in.) OF THE SHAFT » 20.0
ENTER THE NUMBER OF BEARINGS » 2
ENTER THE DISTANCE(in.) TO BEARING(1) » 1.0
ENTER THE DISTANCE(in.) TO BEARING(2) » 19.0

Fig. 3-2. Question Screen 2

At the completion of these questions the software graphically displays the design area as shown in figure 3-3. Here the length of the shaft and bearing locations are shown with a reference set of coordinate axes. At the finish of all graphical displays the user must press the space bar to resume normal execution.
After the shaft length and bearing locations are known the next question screen prompts the user for the hub and loading information, as shown in figure 3-4. While supplying this information keep in mind the sense of the direction of the applied forces which is a positive load acts down, a positive torque follows the right hand rule and a positive axial thrust acts to the right along the axis of the shaft.
*** HUB & LOAD INFORMATION ***

ENTER THE NUMBER OF HUBS >> 3

ENTER THE DISTANCE (in.) TO HUB(1) >> 4.25
ENTER THE WIDTH (in.) OF HUB(1) >> 1.25
ENTER THE CONC.LOAD(lbs) ON HUB(1) >> 1700.0
ENTER THE TORQUE(in-lbs) ON HUB(1) >> 2000.0
ENTER THE THRUST(lbs) ON HUB(1) >> -117.0

ENTER THE DISTANCE (in.) TO HUB(2) >> 10.0
ENTER THE WIDTH (in.) OF HUB(2) >> 2.3
ENTER THE CONC.LOAD(lbs) ON HUB(2) >> -2100.0
ENTER THE TORQUE(in-lbs) ON HUB(2) >> -3500.0
ENTER THE THRUST(lbs) ON HUB(2) >> 300.0

ENTER THE DISTANCE (in.) TO HUB(3) >> 15.0
ENTER THE WIDTH (in.) OF HUB(3) >> 1.0
ENTER THE CONC.LOAD(lbs) ON HUB(3) >> 1113.0
ENTER THE TORQUE(in-lbs) ON HUB(3) >> 1500.0
ENTER THE THRUST(lbs) ON HUB(3) >> -97.0

Fig. 3-4. Question Screen for the Hub and Load Information

When all the information pertaining to the hubs and the loading has been entered, the software begins to calculate the minimum diameter based on a deflection constraint of 0.01 inches. Figure 3-5 shows the messages which appear on screen to notify the user of the adjustments made to the shaft diameter.
PERFORMING CALCULATIONS --- PLEASE WAIT.

STARTING DIAMETER = 1.0000

ITERATION 1
MAX. DEFLECTION = 0.02230
Diameter 1 = 1.1875
Increase Diameter = 1.3125
DEFLECTION OF SHAFT EXCEEDS 0.01

ITERATION 2
MAX. DEFLECTION = 0.01494
Diameter 1 = 1.3125
Increase Diameter = 1.4375
DEFLECTION OF SHAFT EXCEEDS 0.01

ITERATION 3
MAX. DEFLECTION = 0.01039
Diameter 1 = 1.4375
Increase Diameter = 1.5000
DEFLECTION OF SHAFT EXCEEDS 0.01

ITERATION 4
MAX. DEFLECTION = 0.00876

CALCULATIONS FINISHED, PRESS SPACE BAR TO CONTINUE.

Fig. 3-5. Diameter Adjustment Messages
At the completion of the calculations a beeping signal notifies the user the program is prepared to resume interaction and a message telling the user to continue appears on the screen. At this point the user is given the opportunity to get a hardcopy of the loading, deflection, bending and torque data. Following this option, the user is asked whether a hardcopy of the shear diagram is desired, as shown in figure 3-6.
The moment diagram immediately follows the shear diagram and again the user may output a hardcopy, as shown in figure 3-7.
The bearing selection screen shown in figure 3-8, asks the user for pertinent bearing information. The user must specify whether the inner or outer bearing race rotates, the speed of rotation in revolutions per minute, the bearing design life in hours, the bearing reliability (i.e. 0.99) and filename of the bearing catalog. Following these responses the user may select either roller or ball bearings. After supplying this information the bearing catalog is opened and the most suitable bearing is selected according to the calculated load rating and the required
diameter. The bearing vendor and part number appears on the screen notifying the user of the selection.

*** BEARING SELECTION ***
WHICH RACE ROTATES (I)INNER OR (O)UTER >> i
ENTER THE SPEED OF ROTATION (RPM) >> 900.0
ENTER THE DESIGN LIFE (HOURS) >> 4500.0
ENTER THE BEARING RELIABILITY >> 0.99
ENTER THE BEARING CATALOG FILENAME >> blbear2
IS BEARING(1) (R)OLLER OR (B)ALL ? >> B
IS BEARING(2) (R)OLLER OR (B)ALL ? >> B
BEARING SELECTED : VENDOR: AFBMA PART NUM: 0240
BEARING LOAD RATING >> 5062.50
CALCULATED LOAD RATING >> 1780.49
PRESS SPACE BAR TO CONTINUE ...

Fig. 3-8. Bearing Selection Screen

Following the bearing selection the user is prompted for the tenative locations of grooves, keyways and holes as shown in figure 3-9.
THE FOLLOWING QUESTIONS ARE FOR THE TENATIVE LOCATIONS OF HOLES, GROOVES, AND KEWAYS

ENTER THE NUMBER OF HOLES >> 0
ENTER THE NUMBER OF KEYWAYS >> 3
ENTER THE LOCATION OF KEYWAY(1) >> 4.25
ENTER THE LOCATION OF KEYWAY(2) >> 10.0
ENTER THE LOCATION OF KEYWAY(3) >> 15.0
ENTER THE NUMBER OF GROOVES >> 2
ENTER THE LOCATION OF GROOVE(1) >> 0.7
ENTER THE LOCATION OF GROOVE(2) >> 19.3

Fig. 3-9. Location of Grooves, Keyways and Holes

Once the groove, hole and keyway locations are known, the assembly process is determined so that steps can be calculated to properly seat the hubs and bearings. A layout of the shaft is drawn with the bearings, hubs, grooves, keyways and holes labelled accordingly, as depicted in figure 3-10. To determine the assembly of the components on the shaft, the location of thickest section of the shaft must be known. To facilitate this selection, sections of the shaft are numbered so the user may choose a number for the desired section. The user must also supply the fillet radii of the steps.
Fig. 3-10. Preliminary Design Layout

After the user has specified the thickest section of the shaft, the shaft diameters and section lengths for stepping the shaft are determined. The shaft diameters and section lengths are stored in memory and a graph of the quantity M/EI is displayed on the screen, as shown in figure 3-11. This graph shows the user the effects of the steps on the bending moment data and reveals the areas of maximum bending.
The next question screen, shown in figure 3-12, prompts the user for the specific data on the grooves, holes and keyways. For example, the user is asked the radius of the groove or the diameter of the hole. This data is used in the empirical formulas derived for stress concentration and a strength reduction factor, $K_e$, is determined. Appendix B contains the empirical relations and the raw data used in the determination of the strength reduction factor.
**** STRESS CONCENTRATION ****

DO YOU WANT TO CHANGE THE NUMBER OR LOCATION OF GROOVES ?
ANSWER Y OR N >> n

ENTER RADIUS OF GROOVE(1) >> 0.0625
THE DIAMETER AT THIS SECTION = 1.5750
ENTER INNER DIAMETER OF GROOVE(1) >> 1.375

ENTER RADIUS OF GROOVE(2) >> 0.0625
THE DIAMETER AT THIS SECTION = 1.5750
ENTER INNER DIAMETER OF GROOVE(2) >> 1.375

DO YOU WANT TO CHANGE THE NUMBER OR LOCATION OF KEYWAYS ?
ANSWER Y OR N >> n
IS THE BRINELL HARDNESS OF THE MATERIAL 1) < 200 2) > 200 ?
CHOICE >> 1

DO YOU WANT TO CHANGE THE NUMBER OR LOCATION OF HOLES ?
ANSWER Y OR N >> n

Fig. 3-12. Effects of Stress Concentration

The last screen prompts the user for information on the endurance limit modifying factors, as shown in figure 3-13. These factors take into account the working conditions the mechanical part undergoes while in service. The surface factor, Ka, is based on the surface roughness measured in microinches. The user may wish to refer to table 1-3 to select a surface roughness value. The size factor, Kb, is determined automatically and is based on the shaft diameter at the maximum bending moment effect. The reliability factor, Kc, will be computed from the user's selection of the choices listed on the screen. The strength reduction factor, Ke, is calculated automatically from the worst effects of stress concentration. And lastly the safety
factor, n, is user supplied, a typical value being 1.25. After responding to the final prompt the minimum diameter for static and fatigue failure is calculated.

ENDURANCE-LIMIT MODIFYING FACTORS KA, KB, KC, KE

THE SURFACE FACTOR (KA)
IS BASED ON THE SURFACE ROUGHNESS MEASURED IN MICROINCHES
ENTER THE SURFACE ROUGHNESS \( \gg 32.0 \)
KA \( \gg 0.9319 \)

THE SIZE FACTOR KB \( \gg 0.7500 \)
THIS BASED ON THE DIAMETER AT THE MAX. MOMENT

ENTER THE NUMBER OF DESIRED RELIABILITY, KC
1) 0.5 2) 0.90 3) 0.95 4) 0.99 5) 0.999
CHOICE \( \gg 4 \)

BASED ON EFFECTS OF STRESS CONCENTRATION
THE ENDURANCE-LIMIT STRENGTH REDUCTION FACTOR KE \( \gg 0.5115 \)

ENTER THE DESIRED FACTOR OF SAFETY N \( \gg 1.25 \)

Fig. 3-13. Endurance Limit Modifying Factors

If the design passes the static and fatigue criteria based on the preliminary design information, the user is notified on the screen and the final preliminary design is output with the design dimensions as shown in figure 3-14a and 3-14b.
STATIC REQUIREMENTS HAVE BEEN SATISFIED.

FATIGUE REQUIREMENTS HAVE BEEN SATISFIED BY A DIA. = 2.0610

PRELIMINARY DESIGN DISPLAY

DO YOU WANT A HARDCOPY OF THE DISPLAY Y/N?

ANSWER >> y

Fig. 3-14a. Final Preliminary Design

<table>
<thead>
<tr>
<th>SECTION NO.</th>
<th>DIAMETER(in)</th>
<th>SECTION LENGTH(in)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.4500</td>
<td>0.6455</td>
</tr>
<tr>
<td>2</td>
<td>1.5750</td>
<td>0.7090</td>
</tr>
<tr>
<td>3</td>
<td>1.8110</td>
<td>2.3330</td>
</tr>
<tr>
<td>4</td>
<td>1.9360</td>
<td>1.1250</td>
</tr>
<tr>
<td>5</td>
<td>2.0610</td>
<td>4.5375</td>
</tr>
<tr>
<td>6</td>
<td>2.1860</td>
<td>1.3000</td>
</tr>
<tr>
<td>7</td>
<td>2.3110</td>
<td>3.8500</td>
</tr>
<tr>
<td>8</td>
<td>2.0610</td>
<td>1.0000</td>
</tr>
<tr>
<td>9</td>
<td>1.8110</td>
<td>3.1455</td>
</tr>
<tr>
<td>10</td>
<td>1.5750</td>
<td>0.7090</td>
</tr>
<tr>
<td>11</td>
<td>1.4500</td>
<td>0.6455</td>
</tr>
</tbody>
</table>

Fig. 3-14b. Final Design Dimensions

* NOTE: BEARINGS ARE AFEMA 8240
If all failure criteria are not satisfied the user is notified on the screen and the process is repeated. When the process begins a second time the user is given the choice of two options, as shown in figure 3-15. The first option allows the user to change only the properties of the material and keep the design specifications constant. The second option allows the user to redefine any of the previous design specifications. This process will be repeated until a shaft design can be found which satisfies the failure criteria.

STATIC REQUIREMENTS NOT SATISFIED, THE SECTION OF MAX. TORQUE/I WITH ASSOCIATED MOMENT HAS DIA. = 0.8140 A DIAMETER = 1.3074 IS NEEDED.

FATIGUE REQUIREMENTS NOT SATISFIED, THE SECTION OF MAX. MOMENT/I WITH ASSOCIATED TORQUE HAS A DIA. = 0.6890 A DIAMETER OF ABOUT DIA. = 0.6980 IS NEEDED.

FATIGUE REQUIREMENTS NOT SATISFIED, THE SECTION OF MAX. TORQUE/I WITH ASSOCIATED MOMENT HAS A DIA. = 0.9140 A DIAMETER OF ABOUT DIA. = 1.3158 IS NEEDED.

DO YOU WANT ONLY TO CHANGE THE ULTIMATE AND/OR TENSILE STRENGTH OF THE MATERIAL?
ANSWER Y OR N >> Y

ENTER THE YIELD STRENGTH (psi) >> 9.6e4

ENTER THE ULTIMATE TENSILE STRENGTH (psi) >> 1.2e5

Fig. 3-15. Notify the User Design Criteria not met
3.3 Limitations and Future Enhancements

The limitations in a piece of software will usually lead to future enhancements. The first versions of software, in most cases, are rough cuts. The new needs of the software have to be recognized and then incorporated into the latest revision. The following discusses the current needs of the shaft design software.

As a design aid, this Shaft Design software simulates the design process but lacks the ability to perform the design in the same manner as an expert. One way the software can be made to more effectively design as an expert, is to provide the user with an increased number of material choices which will automatically specify the material properties. The software can be modified to select a new material if the material chosen by the user does not satisfy all the failure criteria.

A second way in which the software can perform like an expert is through the incorporation of a cost optimal design algorithm to assess the manufacturing cost of a design for each of the available material choices. All of the designs, with their associated attributes and manufacturing costs, can then be stored in a file as possible design solutions. The algorithm can read through the file containing the designs and select the design which is the least costly or this file can be printed for review and evaluation by the designer.
The bearing selection routine chooses all of the bearings on the basis of the worst loading. The routine, in conjunction with the calculations for the steps in the assembly of the shaft, should be extended to select appropriate bearings based on the stepped geometry of the shaft as well as the calculated bearing load rating. The bearings would be checked for fit due to the diameter and width available on the shaft. The bearings and each of their specifications would be stored in memory and output with the shaft dimensions.

A trade off was made between the execution time and the finite difference matrix size. As the matrix size increases the accuracy of the finite difference method increases and so does the execution time. The calculations effected by this increase are the deflection, shear and bending.

Finally, additional catalogs of standard parts such as gears, pulleys and belts should be created and the shaft design software expanded to incorporate these standard parts in the same fashion that the current software uses the bearing catalog.

The shaft design software principally functions as an aid in the determination of the deflection, shear and moment data and in some cases can produce a feasible solution. With these enhancements incorporated into the software, the software will be capable of simulating the
design process of the expert. By capturing the expertise of the designer on the computer, assessing the possible design solutions will lead to a greater savings of time and cost.
BIBLIOGRAPHY

Crandall, Stephen H.  
1956  
Engineering Analysis. New York,  

Halliday, Kenneth R.  
1985  

Kops, L.  
1971  

Mischke, C.R.  
1978  

Mischke, C.R.  
Umasankar, G.  
1985  

Peterson, R.E.  
1953  

Seireg, A.  
1972  
Shigley, Joseph E.  
Mitchell, Larry D.  
1983


Sines, George  
Waisman, J.L.  
1959


Weiss, Sholom M.  
Kulikowski, C.A.  
1984

APPENDIX A

Flow Chart
MATERIAL INFORMATION:
1) Steel \( E = 30.0 \times 10^6 \)
2) Aluminum \( E = 10.0 \times 10^6 \)

Material Selection => Modulus of Elasticity

- Yield Strength \( (S_y) \)
- Ultimate Tensile Strength \( (S_u) \)

- Length of the Shaft \( (\text{in.}) \)
- Number of Bearings
- Location of Bearings \( (\text{in.}) \)

Display Design Area
Show:
- Length of Shaft
- Bearing Locations
- Reference Axes

- Number of Hubs
- Location of Hubs \( (\text{in.}) \)

For Each Hub:
- Concentrated Load \( (\text{lbs}) \)
- Torque \( (\text{in-lbs}) \)
- Axial Thrust \( (\text{lbs}) \)
Solve for the Deflections

Is the Maximum Deflection < 0.01

YES

Calculate:
- Bearing Reactions
- Bending Moments
- Torque Distribution
- Axial Thrust (placed at compressive end of shaft)

Graphically Display:
- Shear Diagram
- Bending Moment Diagram

NO

Increase Diameter
Hardcopy Option: Forces & Loading on Shaft

Bearing Selection:
- Which Race Rotates (I)nner or (O)uter
- Speed of Rotation (RPM)
- Design Life (Hours)
- Bearing Reliability
- Enter Bearing Catalog Name
- Type of Bearing

- Calculate the Bearing Load Rating
- Open Bearing Catalog and choose the appropriate bearing based on the Load Rating and Minimum Diameter

Tenatively Locate:
Grooves, Keyways & Holes
Graphically Layout the Shaft
Label & Show:
- Locations of the Bearings, Hubs, Grooves, Keyways & Holes
- Find the Thick Section of the Shaft to
  Determine Assembly and Steps
- Fillet Radii

- Determine the Shaft Stepped Geometry
- From the Steps Calculate the Moments of Inertia for each Stepped Section
- Graphically Display the Quantity M/EI

- Ask for Changes in the Number or Location of the Grooves, Keyways & Holes
- Get Specifics on each (i.e. radius of groove)

Determine:
- Endurance Limit Modifying Factors
- Factor of Safety
Check the Failure Criteria
- Static
- Fatigue

Does the Design Satisfy all the Failure Criteria

- Display the Final Design, show the Hubs, Bearings, Keys, Grooves, Holes and Note the Bearings Selected
- Writeout the Shaft Dimensions

Repeat the Design Process:
- Change only the material properties
- Change design parameters

NO

YES
APPENDIX B

Stress Concentration Equations and Raw Data
taken from the Graphs of R.E. Peterson
**Stress concentration effects of steps:**

1) **Bending**
2) **Torsion**

### Bending

<table>
<thead>
<tr>
<th>$D/d$</th>
<th>$r/d$</th>
<th>$Kt$</th>
<th>$Kt = 0.878(r/d)^{-0.318}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.03</td>
<td>2.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>2.29</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.15</td>
<td>1.58</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>1.46</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>1.37</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>1.31</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$D/d$</th>
<th>$r/d$</th>
<th>$Kt$</th>
<th>$Kt = 0.893(r/d)^{-0.295}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.03</td>
<td>2.55</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>2.17</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>1.73</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.15</td>
<td>1.66</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>1.43</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>1.36</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>1.3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$D/d$</th>
<th>$r/d$</th>
<th>$Kt$</th>
<th>$Kt = 0.912(r/d)^{-0.273}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>0.03</td>
<td>2.42</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>2.06</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>1.67</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.15</td>
<td>1.52</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>1.41</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>1.34</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>1.29</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$D/d$</th>
<th>$r/d$</th>
<th>$Kt$</th>
<th>$Kt = 0.927(r/d)^{-0.251}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2</td>
<td>0.03</td>
<td>2.27</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>1.96</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>1.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.15</td>
<td>1.48</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>1.39</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>1.32</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>1.27</td>
<td></td>
</tr>
</tbody>
</table>
## $D/d = 1.1$

<table>
<thead>
<tr>
<th>r/d</th>
<th>Kt</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.03</td>
<td>2.23</td>
</tr>
<tr>
<td>0.05</td>
<td>1.88</td>
</tr>
<tr>
<td>0.1</td>
<td>1.58</td>
</tr>
<tr>
<td>0.15</td>
<td>1.46</td>
</tr>
<tr>
<td>0.2</td>
<td>1.37</td>
</tr>
<tr>
<td>0.25</td>
<td>1.31</td>
</tr>
<tr>
<td>0.3</td>
<td>1.26</td>
</tr>
</tbody>
</table>

$k_t = 0.928(d/r)^{-0.242}$

## $D/d = 1.05$

<table>
<thead>
<tr>
<th>r/d</th>
<th>Kt</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.03</td>
<td>2.01</td>
</tr>
<tr>
<td>0.05</td>
<td>1.78</td>
</tr>
<tr>
<td>0.1</td>
<td>1.53</td>
</tr>
<tr>
<td>0.15</td>
<td>1.41</td>
</tr>
<tr>
<td>0.2</td>
<td>1.34</td>
</tr>
<tr>
<td>0.25</td>
<td>1.29</td>
</tr>
<tr>
<td>0.3</td>
<td>1.25</td>
</tr>
</tbody>
</table>

$k_t = 0.964(d/r)^{-0.206}$

## $D/d = 1.02$

<table>
<thead>
<tr>
<th>r/d</th>
<th>Kt</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.03</td>
<td>1.83</td>
</tr>
<tr>
<td>0.05</td>
<td>1.64</td>
</tr>
<tr>
<td>0.1</td>
<td>1.44</td>
</tr>
<tr>
<td>0.15</td>
<td>1.33</td>
</tr>
<tr>
<td>0.2</td>
<td>1.27</td>
</tr>
<tr>
<td>0.25</td>
<td>1.22</td>
</tr>
<tr>
<td>0.3</td>
<td>1.2</td>
</tr>
</tbody>
</table>

$k_t = 0.945(d/r)^{-0.186}$
### Torsion

#### D/d = 2

<table>
<thead>
<tr>
<th>r/d</th>
<th>Kt</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.02</td>
<td>2.27</td>
</tr>
<tr>
<td>0.03</td>
<td>2.0</td>
</tr>
<tr>
<td>0.05</td>
<td>1.73</td>
</tr>
<tr>
<td>0.1</td>
<td>1.45</td>
</tr>
<tr>
<td>0.15</td>
<td>1.33</td>
</tr>
<tr>
<td>0.2</td>
<td>1.26</td>
</tr>
<tr>
<td>0.25</td>
<td>1.21</td>
</tr>
<tr>
<td>0.3</td>
<td>1.18</td>
</tr>
</tbody>
</table>

#### D/d = 1.33

<table>
<thead>
<tr>
<th>r/d</th>
<th>Kt</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.02</td>
<td>2.15</td>
</tr>
<tr>
<td>0.03</td>
<td>1.93</td>
</tr>
<tr>
<td>0.05</td>
<td>1.68</td>
</tr>
<tr>
<td>0.1</td>
<td>1.41</td>
</tr>
<tr>
<td>0.15</td>
<td>1.29</td>
</tr>
<tr>
<td>0.2</td>
<td>1.22</td>
</tr>
<tr>
<td>0.25</td>
<td>1.18</td>
</tr>
<tr>
<td>0.3</td>
<td>1.15</td>
</tr>
</tbody>
</table>

#### D/d = 1.2

<table>
<thead>
<tr>
<th>r/d</th>
<th>Kt</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.02</td>
<td>1.99</td>
</tr>
<tr>
<td>0.03</td>
<td>1.79</td>
</tr>
<tr>
<td>0.05</td>
<td>1.57</td>
</tr>
<tr>
<td>0.1</td>
<td>1.33</td>
</tr>
<tr>
<td>0.15</td>
<td>1.23</td>
</tr>
<tr>
<td>0.2</td>
<td>1.17</td>
</tr>
<tr>
<td>0.25</td>
<td>1.14</td>
</tr>
<tr>
<td>0.3</td>
<td>1.12</td>
</tr>
</tbody>
</table>

#### D/d = 1.09

<table>
<thead>
<tr>
<th>r/d</th>
<th>Kt</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.02</td>
<td>1.54</td>
</tr>
<tr>
<td>0.03</td>
<td>1.4</td>
</tr>
<tr>
<td>0.05</td>
<td>1.27</td>
</tr>
<tr>
<td>0.1</td>
<td>1.17</td>
</tr>
<tr>
<td>0.15</td>
<td>1.13</td>
</tr>
<tr>
<td>0.2</td>
<td>1.11</td>
</tr>
<tr>
<td>0.25</td>
<td>1.09</td>
</tr>
<tr>
<td>0.3</td>
<td>1.08</td>
</tr>
</tbody>
</table>

Kt = 0.856(r/d)\(^{-0.242}\)

Kt = 0.843(r/d)\(^{-0.235}\)

Kt = 0.834(r/d)\(^{-0.216}\)

Kt = 0.886(r/d)\(^{-0.132}\)
Stress concentration effects of grooves:

Bending

<table>
<thead>
<tr>
<th>D/d = 1.02</th>
<th>r/d</th>
<th>Kt</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>1.66</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>1.44</td>
<td></td>
</tr>
<tr>
<td>0.15</td>
<td>1.35</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>1.29</td>
<td></td>
</tr>
<tr>
<td>0.25</td>
<td>1.25</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>1.22</td>
<td></td>
</tr>
</tbody>
</table>

\[ K_t = 0.983(r/d) - 0.171 \]

<table>
<thead>
<tr>
<th>D/d = 1.05</th>
<th>r/d</th>
<th>Kt</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>1.9</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>1.59</td>
<td></td>
</tr>
<tr>
<td>0.15</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>1.36</td>
<td></td>
</tr>
<tr>
<td>0.25</td>
<td>1.29</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>1.25</td>
<td></td>
</tr>
</tbody>
</table>

\[ K_t = 0.938(r/d) - 0.235 \]

<table>
<thead>
<tr>
<th>D/d = 1.15</th>
<th>r/d</th>
<th>Kt</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>2.14</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>1.7</td>
<td></td>
</tr>
<tr>
<td>0.15</td>
<td>1.53</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>1.42</td>
<td></td>
</tr>
<tr>
<td>0.25</td>
<td>1.34</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>1.28</td>
<td></td>
</tr>
</tbody>
</table>

\[ K_t = 0.899(r/d) - 0.284 \]

<table>
<thead>
<tr>
<th>D/d = 1.5</th>
<th>r/d</th>
<th>Kt</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>2.25</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>1.75</td>
<td></td>
</tr>
<tr>
<td>0.15</td>
<td>1.56</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>1.44</td>
<td></td>
</tr>
<tr>
<td>0.25</td>
<td>1.35</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>1.28</td>
<td></td>
</tr>
</tbody>
</table>

\[ K_t = 0.873(r/d) - 0.311 \]
## Torsion

<table>
<thead>
<tr>
<th>$D/d = 1.3$</th>
<th>$r/d$</th>
<th>$K_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.05</td>
<td>1.77</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>1.48</td>
</tr>
<tr>
<td></td>
<td>0.15</td>
<td>1.37</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>1.3</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>1.25</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>1.21</td>
</tr>
</tbody>
</table>

$K_t = 0.929(r/d)^{-0.209}$

<table>
<thead>
<tr>
<th>$D/d = 1.05$</th>
<th>$r/d$</th>
<th>$K_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.05</td>
<td>1.53</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>1.35</td>
</tr>
<tr>
<td></td>
<td>0.15</td>
<td>1.27</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>1.22</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>1.19</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>1.17</td>
</tr>
</tbody>
</table>

$K_t = 0.963(r/d)^{-0.151}$

<table>
<thead>
<tr>
<th>$D/d = 1.02$</th>
<th>$r/d$</th>
<th>$K_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.05</td>
<td>1.39</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>1.27</td>
</tr>
<tr>
<td></td>
<td>0.15</td>
<td>1.21</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>1.18</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>1.15</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>1.14</td>
</tr>
</tbody>
</table>

$K_t = 0.986(r/d)^{-0.112}$
**Transverse Hole**

**Bending**

<table>
<thead>
<tr>
<th>( \frac{d}{D} )</th>
<th>( K_t )</th>
<th>( K_t = 1.64(\frac{d}{D})^{-0.131} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.02</td>
<td>2.65</td>
<td></td>
</tr>
<tr>
<td>0.04</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td>0.06</td>
<td>2.4</td>
<td></td>
</tr>
<tr>
<td>0.08</td>
<td>2.34</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>2.25</td>
<td></td>
</tr>
<tr>
<td>0.12</td>
<td>2.2</td>
<td></td>
</tr>
<tr>
<td>0.14</td>
<td>2.15</td>
<td></td>
</tr>
<tr>
<td>0.16</td>
<td>2.1</td>
<td></td>
</tr>
<tr>
<td>0.18</td>
<td>2.06</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>2.03</td>
<td></td>
</tr>
<tr>
<td>0.22</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>0.24</td>
<td>1.96</td>
<td></td>
</tr>
<tr>
<td>0.26</td>
<td>1.94</td>
<td></td>
</tr>
<tr>
<td>0.28</td>
<td>1.92</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>1.88</td>
<td></td>
</tr>
</tbody>
</table>

**Torsion**

<table>
<thead>
<tr>
<th>( \frac{d}{D} )</th>
<th>( K_t )</th>
<th>( K_t = 1.14(\frac{d}{D})^{-0.107} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.02</td>
<td>1.74</td>
<td></td>
</tr>
<tr>
<td>0.04</td>
<td>1.63</td>
<td></td>
</tr>
<tr>
<td>0.06</td>
<td>1.54</td>
<td></td>
</tr>
<tr>
<td>0.08</td>
<td>1.49</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>1.45</td>
<td></td>
</tr>
<tr>
<td>0.12</td>
<td>1.42</td>
<td></td>
</tr>
<tr>
<td>0.14</td>
<td>1.39</td>
<td></td>
</tr>
<tr>
<td>0.16</td>
<td>1.38</td>
<td></td>
</tr>
<tr>
<td>0.18</td>
<td>1.36</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>1.35</td>
<td></td>
</tr>
<tr>
<td>0.22</td>
<td>1.34</td>
<td></td>
</tr>
<tr>
<td>0.24</td>
<td>1.33</td>
<td></td>
</tr>
<tr>
<td>0.26</td>
<td>1.32</td>
<td></td>
</tr>
<tr>
<td>0.28</td>
<td>1.31</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>1.31</td>
<td></td>
</tr>
</tbody>
</table>
Keyway stress concentration factors taken from Juvinall

<table>
<thead>
<tr>
<th>Hardness Brinell</th>
<th>Bending</th>
<th>Torsion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hb ≤ 200</td>
<td>Kt = 1.6</td>
<td>Kt = 1.3</td>
</tr>
<tr>
<td>Hb &gt; 200</td>
<td>Kt = 2.0</td>
<td>Kt = 1.6</td>
</tr>
</tbody>
</table>
APPENDIX C

Source Code: Database Manager & Shaft Design
Language: Turbo-Pascal
PROGRAM Bear_Mangr(input,output);

CONST
  MAX = 50;
  Max_Char = 6;
  Bl = ' ';
  Ftype = '.DAT';

TYPE
  Strings = String[Max_Char];
  String12= String[12];

VAR
  VND,Prt_Num,s : Strings;
  BORE,OD,WB,FR,DS,DH,CR : Real;
  Choice : Char;
  UserQuits: Boolean;
  Filename,TempFile : String12;
  Catalog,Outfile,Printer : Text;
  i : Integer;

PROCEDURE Lineditor;

LABEL
  1,2,3;

VAR
  i,j,k,l,Cnt,No_Entries,No_Delete : Integer;
  Answer,Reply,Respon,Option,Quit,Unit : Char;
  CNVL,CNVF : Real;

BEGIN
  ClrScr;
  Writeln(Bl:23, '**** LINE EDITOR ****');
  Writeln;
  Write(Bl:23, rENTER THE CATALOG NAME » .);
  Readln( Filename );
  FOR l:= 1 TO Length( Filename ) DO
    Filename[l]:= UpCase( Filename[l] );
  Filename:= Concat( Filename,Ftype );
  Writeln;
  Assign( Catalog,Filename );
  Reset( Catalog );
  Assign( Outfile,'TEMP.DAT');
  Rewrite( Outfile );
  ClrScr;

  j:=0;
  i:=0;
  No_Delete:= 0;
WHILE NOT Eof( Catalog ) DO
BEGIN
  Readln(Catalog,VND,Prt_Num,BORE,OD,WB,FR,DS,DH,CR);
  j:= j + 1;
END; {while}
Close( Catalog );
No_Entries:= j;
Reset( Catalog );

ClrScr;
REPEAT
  Writeln('ENTER DATA UNITS (E)NGLISH-inch OR (M)ETRIC-millimeter');
  Write('CHOOSE EITHER E OR M » ');
  Readln(Unit); Unit := UpCase(Unit);
  CASE Unit OF
    'E':
      BEGIN
        CNVL := 1.0;
        CNVF := 1.0;
      END;
    'M':
      BEGIN
        CNVL := 25.4;
        CNVF := 0.225;
      END
    ELSE Writeln('ERROR : SELECTION IS INVALID. PLEASE TRY AGAIN');
      END
  UNTIL ( Unit= 'E' ) OR Unit= 'M');

WHILE NOT Eof( Catalog ) DO
BEGIN
  IF i > 0 THEN
    BEGIN
      Writeln;
      Write('DO YOU WANT TO QUIT EDITING? Y/N >> ');
      Readln( Quit ); Quit:= UpCase( Quit );
      IF Quit = 'Y' THEN GOTO 2;
    END;
  1: Readln(Catalog,VND,Prt_Num,BORE,OD,WB,FR,DS,DH,CR);
  i:= i + 1;
 ClrScr;
  Writeln('ENTRY No. ',i,' CATALOG: ',Filename);
REPEAT
Writeln;
Writeln('DO YOU WANT TO DELETE THIS LINE?');
Write('ANSWER Y OR N >> ');
Readln(Reply); Reply:= UpCase(Reply);
IF Reply = 'Y' THEN
BEGIN
Writeln('ARE YOU SURE? Y/N >> ');
Readln(Respon); Respon:= UpCase(Respon);
IF Respon = 'Y' THEN
BEGIN
No_Delete:= No_Delete + 1;
IF (Respon = 'Y') AND (i < No_Entries) THEN GOTO 1;
IF (Respon = 'Y') AND (i = No_Entries) THEN GOTO 3;
END; {if}
END; {if}
IF (Reply <> 'Y') AND (Reply <> 'N') THEN
Writeln('ERROR: INVALID RESPONSE -- TRY AGAIN');
UNTIL (Reply = 'Y') OR (Reply = 'N');
REPEAT
Writeln;
Writeln('DO YOU WANT TO CHANGE THESE BEARING SPECS?');
Write('ANSWER Y OR N >> ');
Readln(Answer); Answer:= UpCase(Answer);
CASE Answer OF
'Y': BEGIN
Writeln; Write('CHANGE THE VENDOR? Y/N >> ');
Readln(Option); Option:= UpCase(Option);
IF Option = 'Y' THEN
BEGIN
Writeln; Write('ENTER VENDOR NAME (6 CHAR. MAX.) >> ');
Readln(VND);
FOR l:= 1 TO Length(VND) DO
VND[l]:= UpCase(VND[l]);
IF Length(VND) < Max_Char THEN
BEGIN
Cnt:= Length(VND) + 1;
FOR l:= Cnt TO Max_Char DO
VND:= Concat(VND,' ');
END; {if}
END; {if}
Write('CHANGE THE PART NUMBER? Y/N >> ')

Readln( Option );  Option:= UpCase(Option);

IF Option = 'Y' THEN
BEGIN
Write('ENTER PART NUMBER ( 6 CHAR. MAX. ) >> ');
Readln( Prt_Num );
IF Length(Prt_Num) < Max_Char THEN
BEGIN
Cnt:= Length(Prt_Num) + 1;
FOR l:= Cnt TO Max_Char DO
Prt_Num:= Concat( Prt_Num, '
' );
END; {if}
END; {if}
Write('CHANGE THE BORE DIAMETER ? Y/N >> ');
Readln( Option );  Option:= UpCase(Option);

IF Option = 'Y' THEN
BEGIN
Write('ENTER THE BORE DIAMETER >> ');
Readln( BORE );
BORE:= BORE/CNVL;
END; {if}
Write('CHANGE THE OUTSIDE DIAMETER ? Y/N >> ');
Readln( Option );  Option:= UpCase(Option);

IF Option = 'Y' THEN
BEGIN
Write('ENTER THE OUTSIDE DIAMETER >> ');
Readln( OD );
OD:= OD/CNVL;
END; {if}
Write('CHANGE THE BEARING WIDTH ? Y/N >> ');
Readln( Option );  Option:= UpCase(Option);

IF Option = 'Y' THEN
BEGIN
Write('ENTER THE BEARING WIDTH >> ');
Readln( WB );
WB:= WB/CNVL;
END; {if}
Write('CHANGE THE FILLET RADIUS ? Y/N >> ');
Readln( Option );  Option:= UpCase(Option);
IF Option = 'Y' THEN
BEGIN
    Write('ENTER THE FILLET RADIUS >> ');
    Readln( FR );
    FR:= FR/CNVL;
END; {if}
Write('CHANGE THE DIAMETER SHOUL. ? Y/N

Option );

IF Option = 'Y' THEN
BEGIN
    Write('ENTER THE DIAMETER SHOUL. >> ');
    Readln( DS );
    DS:= DS/CNVL;
END; {if}
Write('CHANGE THE DIAMETER HUB ? Y/N

Option );

IF Option = 'Y' THEN
BEGIN
    Write('ENTER THE DIAMETER HUB >> ');
    Readln( DH );
    DH:= DH/CNVL
END; {if}
Write('CHANGE THE LOAD RATING ? Y/N

Option );

IF Option = 'Y' THEN
BEGIN
    Write('ENTER THE LOAD RATING >> ');
    Readln( CR );
    CR:= CR*CNVF;
END; {if}

Writeln(Outfile,VND,Prt_Num,BORE,OD,WB,FR,DS,DH,CR);
END; {case y}

'N': BEGIN

Writeln(Outfile,VND,Prt_Num,BORE,OD,WB,FR,DS,DH,CR);
END; {case n}
ELSE Writeln('ERROR: INVALID RESPONSE -- TRY AGAIN');
END {case}
UNTIL ( Answer = 'Y' ) OR ( Answer = 'N' );
END; {while loop}
GOTO 3;
2: FOR k:= i TO No_Entries DO
BEGIN

Readln(Catalog,VND,Prt_Num,BORE,OD,WB,FR,DS,DH,CR);

Writeln(Outfile,VND,Prt_Num,BORE,OD,WB,FR,DS,DH,CR);
END; {for}
3: Writeln; Writeln(Bl:22,'END OF CATALOG, EDITING COMPLETE');
Close( Catalog );
Close( Outfile );
Reset( Outfile );
Rewrite( Catalog );
FOR k:= 1 TO No_Entries - No_Delete DO
BEGIN
Readln(Outfile,VND,Prt_Num,BORE,OD,WB,FR,DS,DH,CR);
Writeln(Catalog,VND,Prt_Num,BORE,OD,WB,FR,DS,DH,CR);
END; {for}
Close( Catalog );
Close( Outfile );
Erase( Outfile );
END; {procedure lineditor}

PROCEDURE Enter_Data;

VAR
 i,j,l,Cnt : Integer;
Unit,ans : Char;
CNVL,CNVF : Real;
Exit : Boolean;
Vendor_Num : Char;

BEGIN
ClrScr;
REPEAT
Writeln('ENTER DATA UNITS (E)NGLISH-inch OR (M)ETRIC-millimeter');
Write('CHOOSE EITHER E OR M >> ');
Readln(Unit); Unit := UpCase(Unit);
CASE Unit OF
'E':
 BEGIN
  CNVL := 1.0;
  CNVF := 1.0;
 END;
'M':
 BEGIN
  CNVL := 25.4;
  CNVF := 0.225;
 END;
END;

ELSE Writeln('ERROR : SELECTION IS INVALID. PLEASE TRY AGAIN');
END
UNTIL ( Unit= 'E' ) OR ( Unit= 'M' );
Writeln;
Exit := False;
REPEAT
   Write('MAKE A DATA ENTRY? Y/N >> ');
   Readln(ans); ans := UpCase(ans);
   CASE ans OF
      'Y':
      BEGIN
         ClrScr;
         Writeln('VENDOR 1. SKF 2. AFBMA 3. TIMKEN 4. TORRINGTON 5. OTHER');
         Write('ENTER NUMBER OF SELECTION >> ');
         Readln(Vendor_Num);
         CASE Vendor_Num OF
            '1': VND := 'SKF ';
            '2': VND := 'AFBMA ';
            '3': VND := 'TIMKEN';
            '4': VND := 'TORRIN';
            '5': BEGIN
               Write('ENTER THE VENDOR NAME ( 6 CHAR. MAX ) >> ');
               Readln(VND);
               FOR j:= 1 TO Length(VND) DO VND[j]:= UpCase(VND[j]);
               IF Length(VND) < Max_Char THEN
               BEGIN
                  Cnt:= Length(VND) + 1;
                  FOR 1:= Cnt TO Max_Char DO
                     VND:= Concat( VND, ' ');
               END;
            END;
         END;
      END; (option 5)
      ELSE Writeln('ERROR : INVALID CHOICE');
      END; (case of)
   END; Write('PART NUMBER ( MAX. 6 NUMBERS ) >> ');
   Readln(Prt_Num);
   IF Length(Prt_Num) < Max_Char THEN
      BEGIN
         Cnt:= Length(Prt_Num) + 1;
         FOR 1:= Cnt TO Max_Char DO
            Prt_Num:= Concat( Prt_Num, ' ');
      END; (if)
Readln(BORE);
  BORE := BORE/CNVL;
  Writeln; Write('BORE DIAMETER >> ');
Readln(OD);
  OD := OD/CNVL;
  Writeln; Write('OUTSIDE DIAMETER >> ');
Readln(FR);
  FR := FR/CNVL;
  Writeln; Write('FILLET RADIUS >> ');
Readln(DS);
  DS := DS/CNVL;
  Writeln; Write('SHOULDER DIA.--INNER RACE >> ');
Readln(DH);
  DH := DH/CNVL;
  Writeln; Write('SHOULDER DIA.--OUTER RACE >> ');
Readln(WB);
  WB := WB/CNVL;
  Writeln; Write('WIDTH >> ');
Readln(CR);
  CR := CR*CNVF;
  Writeln; Write('LOAD RATING >> ');
Writeln(Catalog,VND,Prt_Num,BORE,OD,WB,FR,DS,DH,CR);
  ClrScr;
  END;
'N':
  BEGIN
    Exit := True;
    Close(Catalog);
  END
END
UNTIL Exit
END; {procedure enter_data}

PROCEDURE Add_Catalog;

VAR
  i,l : Integer;
BEGIN
  ClrScr;
  Writeln(B1:23,'*** APPEND TO BEARING CATALOG ***');
  Writeln; Writeln;
  Write(B1:26,'ENTER CATALOG FILE NAME >> ');
  Readln(Filename);
  FOR l := 1 TO Length(Filename) DO
    Filename[l] := UpCase(Filename[l]);
  Filename := Concat(Filename,Ftype);
  Assign(Catalog,Filename);
  Reset(Catalog);
  Append( Catalog );
  Enter_Data;
  Close(Catalog);
PROCEDURE UpDate( VAR Cnt:Integer );

BEGIN
  GotoXY(5,25); Write('PRESS ANY KEY TO CONTINUE ...');
  Repeat Until KeyPressed;
  ClrScr; Writeln('CATALOG : ',Filename);
  Writeln('VENDOR/NO.',BL:5,'BORE',BL:7,'OD',BL:7,'WIDTH',-BL:4,'FILLET',BL:3,'DIA-SHOUL',BL:2,'DIA-HUB',BL:3,'LOAD');
  Cnt:= 0;
END; {procedure update}

PROCEDURE Review;

VAR 1,Cnt : Integer;

BEGIN
  Cnt:= 0;
  ClrScr;
  Write(BL:23,'*** REVIEW BEARING CATALOG ***');
  Writeln;
  Write(BL:26,'ENTER THE CATALOG NAME » '); Readln(Filename);
  FOR 1 := 1 to Length(Filename) DO Filename[1] := UpCase(Filename[1]);
  Filename := Concat(Filename,Ftype);
  Writeln; Writeln(BL:23,'CATALOG NAME ',filename);
  Assign(Catalog,Filename);
  Reset(Catalog);
  ClrScr; Writeln('CATALOG : ',Filename);
  Writeln('VENDOR/NO.',BL:5,'BORE',BL:7,'OD',BL:7,'WIDTH',-BL:4,'FILLET',BL:3,'DIA-SHOUL',BL:2,'DIA-HUB',BL:3,'LOAD');
  WHILE NOT Eof(Catalog) DO BEGIN
    Readln(Catalog,VND,Prt_Num,BORE,OD,WB,FR,DS,DH,CR);
    Cnt:= Cnt +1;
    IF Cnt = 11 THEN UpDate( Cnt );
  END;
  GotoXY(5,24);
  Writeln('PRESS ANY KEY TO CONTINUE...');
  Repeat Until KeyPressed;
  Close(Catalog);
END; {review procedure}

PROCEDURE Print;

VAR 1 : Integer;
BEGIN
ClrScr;
Writeln(B1:23,'*** PRINT BEARING CATALOG ***');
Write(B1:23,'ENTER NAME OF CATALOG TO PRINT >> ');
Readln(Filename);
FOR 1 := 1 TO Length(Filename) DO
Filename[l] := UpCase(Filename[l]);
Filename := Concat(Filename,Ftype);
Assign(Catalog,Filename);
Reset(Catalog);
Writeln; Writeln;
Writeln(Bl:20,'TURN ON PRINTER, PRESS ANY KEY TO CONTINUE ...');
Repeat Until Keypressed;
Writeln; Writeln; Writeln(B1:23,'PRINTING CATALOG : ',Filename);
Assign(Printer,'LST: ');
Rewrite(Printer);
Write(Printer,CHR(27),'N' ,CHR(3) );
Writeln(Printer,Bl:30,'CATALOG ',Filename);
Writeln(Printer);
WHILE NOT Eof(Catalog) DO
BEGIN
Readln(Catalog,VND,Prt_Num,BORE,OD,WB,FR,DS,DH,CR);
Writeln(Printer); Writeln(Printer);
END; {while loop}
Close(Printer);
END; {print procedure}

PROCEDURE Sort;

TYPE
Strings = String[Max_Char];

VAR
VNDT,Prt_NumT : ARRAY[1..MAX] OF Strings;
BORET,ODT,WBT,FRT,DST,DHT,CRT : ARRAY[1..MAX] OF Real;
i,j,1,No_Bearings : Integer;
Temp1,Temp2 : Strings;
Temp3,Temp4,Temp5,Temp6,Temp7,Temp8,Temp9 : Real;

BEGIN
ClrScr;
WriteIn(B1:23,'**** SORT CATALOG ****');
Writeln;
Write(Bl:23,'ENTER THE CATALOG NAME >> ');
Readln(Filename);
FOR I := 1 TO Length(Filename) DO
    Filename[I] := UpCase(Filename[I]);
Filename := Concat(Filename,Ftype);
Writeln; Writeln(Bl:30,'SORTING...');
Assign(Catalog,Filename);
Reset(Catalog);
Assign(Outfile,'TEMP.DAT');
Rewrite(Outfile);
i := 1;
j := 0;
WHILE NOT Eof(Catalog) DO
    BEGIN
        Readln(Catalog,VNDT[i],Prt_NumT[i],BORET[i],ODT[i],WBT[i],-
            FRT[i],DST[i],DHT[i],CRT[i]);
i := i + 1;
j := j + 1;
END; {begin}
Close(Catalog);
No_Bearings := j;
FOR i := 2 TO No_Bearings DO
    FOR j := No_Bearings DOWNTO i DO
        IF CRT[j-1] > CRT[j] THEN
            BEGIN
                Temp1 := VNDT[j-1];
                VNDT[j-1] := VNDT[j];
                VNDT[j] := Temp1;

                Temp2 := Prt_NumT[j-1];
                Prt_NumT[j-1] := Prt_NumT[j];
                Prt_NumT[j] := Temp2;

                Temp3 := BORET[j-1];
                BORET[j-1] := BORET[j];
                BORET[j] := Temp3;

                Temp4 := ODT[j-1];
                ODT[j-1] := ODT[j];
                ODT[j] := Temp4;

                Temp5 := WBT[j-1];
                WBT[j-1] := WBT[j];
                WBT[j] := Temp5;

                Temp6 := FRT[j-1];
                FRT[j-1] := FRT[j];
                FRT[j] := Temp6;

                Temp7 := DST[j-1];
            END;}
DST[j-1] := DST[j];
DST[j] := Temp7;

Temp9 := DHT[j-1];
DHT[j-1] := DHT[j];
DHT[j] := Temp9;

Temp9 := CRT[j-1];
CRT[j-1] := CRT[j];
CRT[j] := Temp9;
END; (loops for sorting)
FOR i := 1 TO No_Bearings DO
BEGIN
  Writeln(Outfile,VNDT[i],Prt_NumT[i],BORET[i],ODT[i],WBT[i]-
  ,FRT[i],DST[i],DHT[i],CRT[i]);
END;
  Close(Outfile);
  Reset(Outfile);
  Rewrite(Catalog);
  FOR i := 1 TO No_Bearings DO
  BEGIN
    Readln(Outfile,VND,Prt_Num,BORE,OD,WB,FR,DS,DH,CR);
    Writeln(Catalog,VND,Prt_Num,BORE,OD,WB,FR,DS,DH,CR);
  END;
  Close(Catalog);
  Close(Outfile);
  Erase(Outfile);
END; (sort procedure)

PROCEDURE Create;

VAR
  ans : Char;
i,l : Integer;

BEGIN (create)
  ClrScr;
  Writeln(B1:26,'*** CREATE BEARING CATALOG ***');
  Writeln;
  Write(B1:22,'ENTER CATALOG NAME ( MAX LENGTH 8 ) >> ');
  Readln(Filename);
  FOR l := 1 TO Length(Filename) DO
    Filename[l] := UpCase(Filename[l]);
  Filename := Concat(Filename,Ftype);
  Writeln;
  Writeln('WARNING : Any file named ',Filename,' will
be destroyed.');
  Write(B1:11,'Do you wish to continue Y/N >> '); Readln(ans); ans := UpCase(ans);
  CASE ans OF
'Y':
    BEGIN
    Assign(Catalog,Filename);
    Rewrite(Catalog);
    Enter_Data;
    END;
'N': Writeln('EXIT CREATE MODE');
END
END; {procedure create}

BEGIN {bear_mangr}
REPEAT
ClrScr;
UserQuits := False;
FOR i := 1 TO Max_Char DO
    BEGIN
    VND[i] := ' ';
    Prt_Num[i] := ' '
    END;
Writeln(B1:22,'*** BEARING CATALOG MANAGER ***');
Writeln; Writeln;
Writeln(B1:32,'MENU :');
Writeln;
Writeln(B1:27,'Create Catalog ... C');
Writeln(B1:27,'Add to Catalog ... A');
Writeln(B1:27,'Review Catalog ... R');
Writeln(B1:27,'Print Catalog .... P');
Writeln(B1:27,'Sort Catalog ..... S');
Writeln(B1:27,'Edit Catalog ..... E');
Writeln(B1:27,'Quit Program ..... Q');
Writeln; Writeln;
Write(B1:27,'SELECT OPTION >> ');
Read(Kbd,Choice); Choice := UpCase(Choice);
CASE Choice OF
    'C' : Create;
    'A' : Add_Catalg;
    'R' : Review;
    'P' : Print;
    'S' : Sort;
    'E' : Lineditor;
    'Q' : UserQuits := TRUE;
ELSE Writeln('ERROR : INVALID SELECTION.
PLEASE TRY AGAIN');
    END;
    UNTIL UserQuits;
END. {bear_mangr}
PROGRAM Deflect(input,output);
{$I typedef.sys}
{$I graphix.sys}
{$I kernel.sys}

LABEL
1,2;

CONST
PI = 3.141592;
B1 = ' ';
MX = 51;  (Number of equations or pts. along shaft)
MAX = 52;  (No. of Eqn +1 Load Vector)
Ftype = '..DAT';
Max_Char = 6;

TYPE
Real2 = ARRAY[1..2] OF Real;
Real11= ARRAY[1..11] OF Real;
Vals = ARRAY[0..MX] OF Real;
Copy = ARRAY[0..MX] OF Real;
RValues = ARRAY[1..10] OF Real;
IVValues = ARRAY[1..10] OF Integer;
Strings = String[Max_Char];
Matrix = ARRAY[1..MX,1..MAX] OF Real;

VAR
ans,Opt,Choice : Char;
Catalog : Text;
Filename: String[12];
VND,Prt_Num : Strings;
Change Prop : Integer;
S_E,F_Safe,Cal_D1,Cal_D2,Tes_D1,Tes_D2 : Real;

Ktbgr,Ktbh,Kbst,KA,KB,KC,KE,Kf,M_Mom,M_Torq,A_Torq,A_Mom,-
Max_S1,Max_S2 : Real;

No_Grv,No_Key,No_Hole,NumH,Point,WhereToStart,Flag : Integer;

BORE,OD,FR,DS,DH,CR,Fillet,APower,D_Ratio,R_Ratio-
,Keytb,RAD,q : Real;

X,Y,R,Temp,Temp1,Temp2,Max_Y,Min_Y,CRD,V,BRel,NSpeed,-
Step1,Step2 : Real;

Mod_Elast,S_Ys,S_Ult,LShaft,DX,Delta_Conc,TFx,DHours,-
Sum_Torq,Sum : Real;

Start,Finish,No_SecChDia,No_Load,Set_Up,Sect_Th,Str_Raise,-
Compr_End : Integer;

Last,II,JJ,Tick,Mark,MatSize,I,J,K,L,Cnt,No_Bear,No_Seg,MS

: Integer;

Bear_Pos : IVValues;
Load_Loc : IVValues;
FUNCTION Sign(Number : Real):Real;
BEGIN
  IF Number = 0.0
  THEN Sign := 1
  ELSE Sign := Abs(Number)/Number
END; {function sign}

FUNCTION Raise(Number,Power : Real):Real;
BEGIN
  IF Number = 0.0
  THEN IF Power = 0.0
       THEN Raise := 1.0
       ELSE Raise := 0.0
  ELSE Raise := Sign(Number)*Exp(Power*Ln(Abs(Number)))
END; {function raise}

PROCEDURE Design_Area;

VAR
  ch : Char;
  X_Offset, X : Real;
  Max_LenX, Max_LenY, Mid_Scr, Tick, Mark, X_ColShaftEnd, Col_X : Integer;
I, S1, S2, S3, Y_Txt, X_ColAxis, Y_RowAxis, Mark_Bear, Line_Len : Integer;

PROCEDURE Arrows;

VAR
    XAxis_Len, HArrowLen, VArrowLen, HTail, VTail : Real;
    Col_X, Col_X1, Row_Y, Row_Y1 : Integer;

BEGIN
    SetLineStyle(0);
    XAxis_Len := X_OffSet + (0.7*X_OffSet);
    HArrowLen := XAxis_Len - (0.2*XAxis_Len);
    HTail := 0.02*(Mid_Scr + 5);
    VArrowLen := Mid_Scr + 5;
    VTail := 0.07*XAxis_Len;
    DrawLine( X_OffSet, Mid_Scr, XAxis_Len, Mid_Scr );
    DrawLine( X_OffSet, Mid_Scr, X_OffSet, Mid_Scr+7 );
    DrawLine( X_OffSet, Mid_Scr+7, X_OffSet-VTail, VArrowLen );
    DrawLine( X_OffSet, Mid_Scr+7, X_OffSet-VTail, VArrowLen );
    DrawLine( XAxis_Len, Mid_Scr, HArrowLen, Mid_Scr+HTail );
    DrawLine( XAxis_Len, Mid_Scr+HTail, X_OffSet+VTail, VArrowLen );
    DrawLine( XAxis_Len, Mid_Scr, HArrowLen, Mid_Scr+HTail );

    Col_X1 := Trunc( 80*X_OffSet/Max_LenX );
    Row_Y1 := 25-Trunc( 25*(Mid_Scr+7)/Max_LenY );
    GotoXY(Col_X1,Row_Y1); Write('Y');
    Col_X := Trunc( 80*XAxis_Len/Max_LenX );
    Row_Y := 25-Trunc( 25*(Mid_Scr-4)/Max_LenY );
    GotoXY(Col_X,Row_Y); Write('X');
END; {procedure arrows}

BEGIN
    InitGraphic;
    Max_LenX := Trunc( LShaft+1 );
    Max_LenY := 80;
    DefineWorld(1,0,0,Max_LenX,Max_LenY);
    SelectWorld(1);
    SelectWindow(1);
    S1 := 0;
    S2 := Max_LenY - Trunc(0.8*Max_LenY);
    DrawSquare( S1, S2, Max_LenX, Max_LenY, false );
    Y_Txt := 25-Trunc( S2*25/Max_LenY );
    GotoXY(30,Y_Txt); WriteLn('*** SHAFT DESIGN ***');

    SetLineStyle(3);
    S3 := Trunc( (Max_LenY - S2)/2 );
    Mid_Scr := Max_LenY - S3;
    X_OffSet := 0.1*Max_LenX;
    DrawLine( S1, Mid_Scr, Max_LenX, Mid_Scr );
DrawLine( X_OffSet, Mid_Scr-20, X_OffSet, Mid_Scr+5 );
DrawLine( LShaft, Mid_Scr-5, LShaft, Mid_Scr+25 );

SetLineStyle(0);
FOR Tick:= 1 TO Max_LenX DO
BEGIN
    Mark := Tick;
    DrawLine( Mark, Mid_Scr+1, Mark, Mid_Scr-1 )
END; {for}

X_ColAxis := Trunc( 80*X_OffSet/Max_LenX );
Y_RowAxis := Trunc( 25*(Mid_Scr+10)/Max_LenY );
GotoXY( X_ColAxis, Y_RowAxis ); Write('0');

X_ColShaftEnd := Trunc( 80*LShaft/Max_LenX );
GotoXY( X_ColShaftEnd, 2 ); Write( LShaft:5:3 );

SetLineStyle(1);
Mark_Bear := Mid_Scr - 8;
Line.Len := S2 + Trunc( 0.5*S2 );
FOR I:= 1 TO No_Bear DO
BEGIN
    X := Dist_Bear[I];
    DrawLine( X, Mark_Bear, X, Mark_Bear+Line.Len );
    Col_X := Trunc( 80*X/Max_LenX );
    IF Col_X = 0 THEN Col_X := 1;
    GotoXY( Col_X, 4 ); Write( X:5:3 );
END; {for}

GotoXY( 1, 2 ); Write('0');
Arrows;

HardCopy( false,1 );

Repeat
    Read( Kbd, ch )
Until ch = ' ';
LeaveGraphic;
END; {procedure design_area}

PROCEDURE In_Matrix;
VAR
    I,J,K : Integer;
BEGIN
    FOR I:= 1 TO MatSize DO
        FOR J:= 1 TO MatSize+1 DO
            Stiff^[I,J] := 0; {load initial zeros into matrix}

    FOR K:= 3 TO MatSize-2 DO {load stiffness matrix as if ends free}
        BEGIN
            Stiff^[K,K-2] := 1;
            Stiff^[K,K-1] := -4;
Stiff^\text{[K,K]} := 6;
Stiff^\text{[K,K+1]} := -4;
Stiff^\text{[K,K+2]} := 1;
END; \ (\text{for})

Stiff^\text{[1,1]} := 2;
Stiff^\text{[1,2]} := -4;
Stiff^\text{[1,3]} := 2;
Stiff^\text{[2,1]} := -2;
Stiff^\text{[2,2]} := 5;
Stiff^\text{[2,3]} := -4;
Stiff^\text{[2,4]} := 1;
Stiff^\text{[MatSize-1,MatSize-3]} := 1;
Stiff^\text{[MatSize-1,MatSize-2]} := -4;
Stiff^\text{[MatSize-1,MatSize-1]} := 5;
Stiff^\text{[MatSize-1,MatSize]} := -2;
Stiff^\text{[MatSize,MatSize-2]} := 2;
Stiff^\text{[MatSize,MatSize-1]} := -4;
Stiff^\text{[MatSize,MatSize]} := 2;

END; \ {\text{procedure \text{In\_Matrix}}}

\text{PROCEDURE \text{Over\_Write};}

\text{VAR}
I,J,Z,Shift : Integer;

\text{BEGIN}
\text{IF No\_Bear > 1 THEN}
\text{BEGIN}
\text{IF ( Bear\_Pos[1] = 0 ) AND ( Bear\_Pos[No\_Bear] < Trunc( (LShaft-DX)/DX ) ) THEN}
\text{BEGIN}
\text{Set\_Up := 0;}
\text{IF No\_Bear > 1 THEN}
\text{BEGIN}
\text{SHIFT := 2;}
\text{FOR I:= 2 TO No\_Bear DO}
\text{BEGIN}
\text{Z := Bear\_Pos[I] - Shift;}
\text{Shift := Shift + 1;}
\text{Stiff^\text{[Z,Z+2]} := 0;}
\text{Stiff^\text{[Z+1,Z+2]} := 1; \ \{\text{CASE 1}\}}
\text{Stiff^\text{[Z+1,Z+3]} := 0;}
\text{Stiff^\text{[Z+2,Z]} := 0;}
\text{Stiff^\text{[Z+2,Z+1]} := 1;}
\text{Stiff^\text{[Z+3,Z+1]} := 0;}
\text{END; \ \{\text{for}\}}
\text{END; \ \{\text{if}\}}
\text{END; \ (if)}
\text{END; \ (if)}
IF ( Bear_Pos[1] = 1 ) AND ( Bear_Pos[No_Bear] >= Trunc((LShaft-DX)/DX) ) THEN
BEGIN
Set_Up := 1;
IF No_Bear > 2 THEN
BEGIN
Shift := 3;
FOR I:= 2 TO No_Bear-1 DO
BEGIN
Z := Bear_Pos[I] - Shift;
Shift := Shift + 1;
Stiff[Z,Z+2] := 0;
Stiff[Z+1,Z+2] := 1;
Stiff[Z+1,Z+3] := 0;
{CASE 2)
Stiff[Z+2,Z] := 0;
Stiff[Z+2,Z+1] := 1;
Stiff[Z+3,Z+1] := 0;
END; {for}
END; {if}
END; {if}

IF ( Bear_Pos[1] = 0 ) AND ( Bear_Pos[No_Bear] >= Trunc((LShaft-DX)/DX) ) THEN
BEGIN
Set_Up := 0;
IF No_Bear > 2 THEN
BEGIN
Shift := 2;
FOR I:= 2 TO No_Bear-1 DO
BEGIN
Z := Bear_Pos[I] - Shift;
Shift := Shift + 1;
Stiff[Z,Z+2] := 0;
Stiff[Z+1,Z+2] := 1;
Stiff[Z+1,Z+3] := 0;
{CASE 3)
Stiff[Z+2,Z] := 0;
Stiff[Z+2,Z+1] := 1;
Stiff[Z+3,Z+1] := 0;
END; {for}
END; {if}
END; {if}

IF ( Bear_Pos[1] > 1 ) AND ( Bear_Pos[No_Bear] < Trunc((LShaft-DX)/DX) ) THEN
BEGIN
Set_Up := 2;
Shift := 1;
FOR I:= 1 TO No_Bear DO
BEGIN
Z := Bear_Pos[I] - Shift;
Shift := Shift + 1;
Stiff[Z,Z+2] := 0;
Stiff[Z+1,Z+2] := 1;
Stiff[Z+1,Z+3] := 0;
{CASE 1)
Stiff[Z+2,Z] := 0;
Stiff[Z+2,Z+1] := 1;
Stiff[Z+3,Z+1] := 0;
END; {for}
END; {if}
END; {if}
Shift := Shift + 1;
Stiff^*[Z,Z+2] := 0;
Stiff^*[Z+1,Z+2] := 1;  {CASE 4}
Stiff^*[Z+1,Z+3] := 0;
Stiff^*[Z+2,Z] := 0;
Stiff^*[Z+2,Z+1] := 1;
Stiff^*[Z+3,Z+1] := 0;
END; {for}
END; {if}

IF ( Bear_Pos[1] = 1 ) AND ( Bear_Pos[No_Bear] < Trunc((LShaft-DX)/DX) ) THEN
BEGIN
Set_Up := 1;
Shift := 3;
FOR I:= 2 TO No_Bear DO
BEGIN
Z := Bear_Pos[I] - Shift;
Shift := Shift + 1;
Stiff^*[Z,Z+2] := 0;
Stiff^*[Z+1,Z+2] := 1;  {CASE 5}
Stiff^*[Z+1,Z+3] := 0;
Stiff^*[Z+2,Z] := 0;
Stiff^*[Z+2,Z+1] := 1;
Stiff^*[Z+3,Z+1] := 0;
END; {for}
END; {if}
END; {End The Main IF Statement}

IF ( Bear_Pos[1] > 1 ) AND ( Bear_Pos[No_Bear] >= Trunc((LShaft-DX)/DX) ) THEN
BEGIN
Set_Up := 2;
Shift := 1;
FOR I:= 1 TO No_Bear-1 DO
BEGIN
Z := Bear_Pos[I] - Shift;
Shift := Shift + 1;
Stiff^*[Z,Z+2] := 0;
Stiff^*[Z+1,Z+2] := 1;  {CASE 6}
Stiff^*[Z+1,Z+3] := 0;
Stiff^*[Z+2,Z] := 0;
Stiff^*[Z+2,Z+1] := 1;
Stiff^*[Z+3,Z+1] := 0;
END; {for}
END; {if}
END; {End The Main IF Statement}

IF ( Bear_Pos[1] = 0 ) OR ( Bear_Pos[1] = 1 ) THEN
BEGIN
82

\begin{verbatim}
Stiff^[1,1] := 5;
Stiff^[1,3] := 1;
Stiff^[2,1] := -4;
Stiff^[2,2] := 6;
END; {if}

IF Bear_Pos[No_Bear] >= Trunc( (LShaft-DX)/DX ) THEN
BEGIN
  Stiff^[MatSize,MatSize] := 5;
  Stiff^[MatSize,MatSize-2] := 1;
  Stiff^[MatSize-1,MatSize] := -4;
  Stiff^[MatSize-1,MatSize-1] := 6;
END; {if}
END; {procedure Over_Write}

PROCEDURE Inertia;

LABEL
  1, 2;

VAR
  I, J, K, L, Off_Set : Integer;

BEGIN

  FOR I := 1 TO No_Load DO
    Load[I] := (Load[I]*Delta_Conc)/(MomI[1]*Mod_Elast);

  CASE Set_Up OF
    0: Off_Set := 1; {depending on the starting end condition}
    1: Off_Set := 2; {the load location in the matrix must}
    ELSE Off_Set := 0; {have some initial offset, even of 0 }
  END; {case}
  J := 1;

  IF ( Set_Up = 0 ) OR ( Set_Up = 1 ) THEN
    BEGIN
      FOR I := 1 TO No_Load DO {find the correct location in the matrix}
        BEGIN {for the load}
          IF J < No_Bear THEN
            BEGIN
              1: IF Dist_Bear[I+1] < Dist_Load[I] THEN
                  BEGIN
                    Off_Set := Off_Set + 1;
                    J := J + 1;
                  END; {if}
            END; {if}
          END; {for the load}
        END; {for I}
    END; {IF ( Set_Up = 0 ) OR ( Set_Up = 1 ) THEN
  END; {PROCEDURE Inertia}
\end{verbatim}
IF ( J < No_Bear ) AND ( Dist_Bear[J+1] < Dist_Load[I] ) THEN GOTO 1;
Load_Loc[I] := Load_Loc[I] - Off_Set;
END; ( if )
END; ( for )
END; ( if )

IF ( Set_Up <> 0 ) AND ( Set_Up <> 1 ) THEN
BEGIN
FOR I:= 1 TO No_Load Do
BEGIN
IF J <= No_Bear THEN
BEGIN
IF Dist_Bear[J] < Dist_Load[I] THEN
BEGIN
Off_Set := Off_Set + 1;
J := J + 1;
END; ( if )
IF ( J <= No_Bear ) AND ( Dist_Bear[J] < Dist_Load[I] ) THEN GOTO 2;
Load_Loc[I] := Load_Loc[I] - Off_Set;
END; ( if )
END; ( for )
END; ( if )

FOR I:= 1 TO No_Load DO
BEGIN
L := Load_Loc[I];
Stiff[L,MatSize+1] := Load[I];
END; ( for )
END; ( procedure inertia )

PROCEDURE Invert;

LABEL 1;
VAR
R, Temp : Real;
I, J, K : Integer;

BEGIN
FOR K:= 1 TO MatSize DO
BEGIN
Temp := Stiff[K,K];
FOR J:= 1 TO MatSize+1 DO
Stiff[K,J] := Stiff[K,J]/Temp;
FOR I:= 1 TO MatSize DO
BEGIN
IF I=K THEN GOTO 1;
BEGIN
   R := Stiff[I,K];
   FOR J := 1 TO MatSize+1 DO
      Stiff[I,J] := Stiff[I,J] - R*Stiff[K,J];
   END;
END; {for}
END; {procedure invert}

PROCEDURE Copy_Deflect;

LABEL 1;
VAR I,J,K : Integer;
BEGIN
   FOR I := 0 TO MX-1 DO
      D[I] := 0;
   I := 1;
   J := -1;
   CASE Bear_Pos[I] OF
   0:
      BEGIN
         (Here the deflections are rearranged and copied)
         D[0] := 0;
         I := 2;
         J := 0;
         END;
   END; {case of}
   BEGIN
      D[0] := 0;
      D[1] := 0;
      J := 1;
      I := 2;
      END;
   END; {case of}
   MS := MatSize;
   IF Bear_Pos[No_Bear] = Trunc(LShaft/DX) THEN MS := MatSize+1;
   IF Bear_Pos[No_Bear] = Trunc((LShaft-DX)/DX) THEN MS := MatSize+2;
   FOR K := 1 TO MS DO
      BEGIN
         J := J + 1;
         IF ( J = Bear_Pos[I] ) AND ( I <= No_Bear ) THEN
            BEGIN
               D[J] := 0;
               J := J + 1;
            END;
         END;
      END;
END;
I := I + 1;
END; {if}
D[J] := Stiff^[K,MatSize+1];
END; {for}
END; {procedure copy_deflect}

PROCEDURE Beep;

VAR
ch : Char;
BEGIN
Sound(1000);
Delay(225);
NoSound;
Delay(125);
Sound(1000);
Delay(225);
NoSound;
Writeln; Writeln('CALCULATIONS FINISHED, PRESS SPACE BAR TO CONTINUE. ');

Repeat
  Read( Kbd, ch );
  Until ch=' '
END; {procedure beep}

PROCEDURE TOR;

VAR
  I, J, L : Integer;
  Sum_Torq : Real;
BEGIN
  FOR I := 0 TO MX-1 DO
    Torque[I] := 0;

  I := 1;
  J := 2;
  Sum_Torq := Torq[1];
  FOR I := 0 TO MX-1 DO
    BEGIN
      IF L < No_Load THEN
        BEGIN
          IF ( I >= Round(Dist_Load[L]/DX)) AND ( I <=
            Round(Dist_Load[L+1]/DX)) THEN Torque[I] := Sum_Torq;
        END; {if}

      IF J <= No_Load THEN
        BEGIN
          \...
        END; {procedure TOR}
  \...

IF I = Round(Dist_Load[J]/DX) THEN
BEGIN
    Sum_Torq:= Sum_Torq + Torq[J];
    J:= J + 1;
    L:= L + 1;
END; {if}
END; {if}
IF I = Round( Dist_Load[No_LoadJ]/DX ) THEN Torque[I]:= Torque[I-1];
END; {for}
END; {procedure tor}

PROCEDURE VShear;

VAR
    I, LL, KK, II, JJ : Integer;
    Const1, TM2, TM1, T, TP1, TP2 : Real;

BEGIN
    {find the shear forces and the }
    Const1 := 2*DX*DX*DX;
    {bearing reactions}
    J:= 1;

    FOR I:= 0 TO MX-1 DO
        Shear[I]:= 0;

    FOR I:= 2 TO MX-3 DO
        BEGIN
            TM2 := D[I-2]*(1);
            SO, WE MUST }
            TM1 := D[I-1]*(2);
            MX-1 }
            T := D[I]*0;
            LAST IS MX-3 }
            TP1 := D[I+1]*(2);
            TP2 := D[I+2]*(1);
            Shear[I] := (-TM2+TM1+T-TP1+TP2)*(
                (Mod_Elast*MomI[J])/Const1 );
            END; {for}

    Shear[1]:= -(2*D[0]+1*D[1]+D[2]*(-2)+1*D[3])*(
        (Mod_Elast*MomI[1])/Const1 );
    IF Round( Dist_Bear[1]/DX ) = 1 THEN
        BEGIN
            Shear[2]:= Shear[3];
            Shear[1]:= Shear[3];
            END;
    Shear[0]:= Shear[1];
    Shear[MX-2]:= -(D[MX-4]*(-1)+D[MX-3]*(2)+D[MX-2]*(-1)+D[MX-1]*(-2))*(
        (Mod_Elast*MomI[J])/Const1 );
IF Round( Dist_Bear[No_Bear]/DX ) = Round( (LShaft-DX)/DX ) THEN
BEGIN
  Shear[MX-3] := Shear[MX-4];
  Shear[MX-2] := Shear[MX-4];
END;
Shear[MX-1] := Shear[MX-2];

LL := 1;
KK := No_Load;

BEGIN
  J := Round( Dist_Load[1]/DX );
  Shear[J] := Shear[J+1];
  LL := 2;
  IF J > 0 THEN
  BEGIN
    FOR K := 0 TO J-1 DO
      Shear[K] := 0;
  END;
END;

IF Dist_Load[No_Load] > Dist_Bear[No_Bear] THEN
BEGIN
  J := Round( Dist_Load[No_Load]/DX );
  Shear[J] := -1*Shear[J-1];
  KK := No_Load -1;
  IF J < MX-1 THEN
  BEGIN
    FOR K := J+1 TO MX-1 DO
      Shear[K] := 0;
  END;
END;

IF LL <= KK THEN
BEGIN
  FOR I := LL TO KK DO
  BEGIN
    J := Round( Dist_Load[I]/DX );
    Shear[J] := Shear[J+1] - Shear[J-1];
  END;
END;

II := 1;
JJ := No_Bear;

BEGIN
  J := Round( Dist_Bear[1]/DX );
  Shear[J] := Shear[J+1];
  II := 2;
IF \( J > 0 \) THEN
BEGIN
    FOR \( K := 0 \) TO \( J-1 \) DO
        Shear\[K]\:= 0;
END; {for}
END; {if}

IF Dist_Bear[No_Bear] \> Dist_Load[No_Load] THEN
BEGIN
    \( J := \text{Round}\left( \frac{\text{Dist}_B\text{ear}[\text{No}_B\text{ear}]}{\text{DX}} \right) \);
    Shear\[J\]:= -1*Shear\[J-1\];
    \( \text{J\(J\) := No_Bear -1;} \)
    IF \( J < \text{MX-1} \) THEN
    BEGIN
        FOR \( K := J+1 \) TO \( \text{MX-1} \) DO
            Shear\[K]\]:= 0;
    END; {for}
END; {if}

IF II \( \leq \text{JJ} \) THEN
BEGIN
    FOR \( I := II \) TO \( JJ \) DO
    BEGIN
        \( J := \text{Round}\left( \frac{\text{Dist}_B\text{ear}[I]}{\text{DX}} \right) \);
        Shear\[J\]:= Shear\[J+1\] - Shear\[J-1\];
    END; {for}
END; {procedure VShear}

PROCEDURE TMoment;

VAR
    Const2, TM1, T, TP1 : Real;
    I, J : Integer;
BEGIN
    Const2 := DX*DX;
    J := 1;

    FOR \( I := 0 \) TO \( \text{MX-1} \) DO
        Moment\[I\]:= 0;
    FOR \( I := 1 \) TO \( \text{MX-2} \) DO
        BEGIN
            TM1 := \( D[I-1] \);
            T := \( D[I]*(-2) \);
            TP1 := \( D[I+1] \);
            Moment\[I\]:= -\( (\text{TM1} + T + TP1 )*\)
                    (\( \text{Mod}_E\text{last}*\text{MomI}[J]/\text{Const2} \));
        END; {for}
    Moment\[0\]:= 0;
IF Round( Dist_Bear[1]/DX ) = 1 THEN Moment[1]:= 0;
Moment[MX-1]:= 0;
IF Round( Dist_Bear[No_Bear]/DX ) = Round( (LShaft -
DX)/DX ) THEN Moment[MX-2]:= 0;
END; {procedure Vmoment}

PROCEDURE HDCopy;

VAR
  Ch : Char;
  Prt : Text;
  I : Integer;
  W : Real;

BEGIN
  ClrScr;
  Writeln('HARDCOPY OF LOAD AND STRESS ON THE SHAFT');
  Writeln;
  Repeat
    Writeln(' TURN ON THE PRINTER, AND PRESS THE SPACE
BAR TO CONTINUE...');
    Read( Kbd,Ch )
    Until Ch = ' ';
    Assign( Prt, 'LST:' );
    Rewrite( Prt );
    Writeln( Prt,B1:12,'*** LOAD AND STRESS DISTRIBUTION
ON THE SHAFT ***');
    Writeln( Prt );
    Writeln( Prt );
    Writeln(Prt,'HUB
'THRUST');
    Writeln( Prt );
    FOR I:= 1 TO No_Load DO
      Writeln(Prt,B1:3,I,B1:7,Dist_Load[I]:5:3,B1:7,-
      CLoad[I]:7:2,B1:6,Torq[I]:7:2,B1:6,Thrust[I]:7:2);
      Writeln( Prt );
      Writeln(Prt,'DIST.(In)',B1:5,'DEFLECTION(In.)',B1:5,-
      'SHEAR(lb)',B1:5,'MOMENT(lb-in)',B1:5,'TORQUE(lb-in)');
      Writeln( Prt );
      FOR I:= 0 TO MX-1 DO
        BEGIN
          W:= I*DX;
          Writeln(Prt,B1:2,W:5:3,B1:9,D[I]:8:6,B1:8,Shear[I]:8:2,-
          B1:8,Moment[I]:8:2,B1:10,Torque[I]:8:2);
          END;
    Close( Prt );
  END; {procedure hardcopy}

PROCEDURE Max_MinSH;

VAR
X, Y, XX, Sfx, Sfy, Bottom, Top, Tick, Mark, Range, Temp1, Temp2 : Real;

BEGIN

    EnterGraphic;
    GotoXY(1,1);
    Writeln(' SHEAR DIAGRAM');
    Writeln;
    Writeln('DO YOU WANT A HARDCOPY OF THE DIAGRAM Y/N ?');
    Write('ANSWER » I');
    Read( ch ); ch:= UpCase( ch );
    IF ch = 'Y' THEN Flag:= 1 ELSE Flag:= 0;
    ClearScreen;
    Bottom:= 0.125*(Min_Y - Max_Y); { assume min_y is zero or negative no absolute value }
    Top:= Abs( Bottom );
    DefineWorld(2, -1, Min_Y+Bottom, LShaft+2, Max_Y+Top);
    SelectWorld(2);
    SelectWindow(1);
    DrawSquare(-1, Min_Y+Bottom, LShaft+2, Max_Y+Top, false);
    SetLineStyle(1);
    DrawLine( 0, Min_Y, LShaft, Min_Y);
    DrawLine( 0, Min_Y, 0, Max_Y);
    DrawLine( 0, 0, LShaft, 0);
    SetLineStyle(0);
    Tick:= LShaft/10;
    XX:= Abs( 0.25*Bottom );

FOR I:= 1 TO 10 DO
    BEGIN
        Mark:= Tick*I;
        DrawLine( Mark, Min_Y, Mark, Min_Y+XX );
    END; {for}

Range:= LShaft+2.3;
Sfx:= 80/Range;

FOR I:= 1 TO 10 DO
    BEGIN
        X:= 0.4+Tick*I;
        IX:= Round(Sfx*X);
        Mark:= Tick*I;
        GotoXY(IX,24);
        Write(Mark:4:1);
    END; {for}

Tick:= Round( (Max_Y-Min_Y)/10 );

FOR I:= 1 TO 10 DO
    BEGIN
Mark := Min Y + Tick*I;
DrawLine( 0, Mark, 0.02*LShaft, Mark);
END;

Range := (Max_Y + Top) - (Min_Y + Bottom);
Sfy := 25/Range;

FOR I := 1 TO 10 DO
BEGIN
    Y := Abs( Bottom ) + I*Tick;
    IY := 25 - Trunc( Sfy*Y );
    Mark := Min_Y + I*Tick;
    GotoXY( 1, IY ); Write( Mark:4:1 );
END;
{for}

L := 1;
K := 1;
II := No_Bear;
JJ := No_Load;
Temp1 := 0;

BEGIN
    X := Dist_Bear[1];
    J := Round(Dist_Bear[1]/DX);
    DrawLine(X,0,X,Shear[J+1]);
    DrawLine(X,Shear[J+1],(J+1)*DX,Shear[J+1]);
    Temp1 := (J + 1)*DX;
    K := K + 1;
    Temp2 := Shear[J+1];
END;

BEGIN
    X := Dist_Load[1];
    J := Round(Dist_Load[1]/DX);
    DrawLine(X,0,X,Shear[J+1]);
    DrawLine(X,Shear[J+1],(J+1)*DX,Shear[J+1]);
    Temp1 := (J + 1)*DX;
    L := L + 1;
    Temp2 := Shear[J+1];
END;

IF Dist_Bear[No_Bear] > Dist_Load[No_Load] THEN
BEGIN
    X := Dist_Bear[No_Bear];
    I := Round(Dist_Bear[No_Bear]/DX);
    DrawLine( (I-1)*DX,Shear[I-1],X,Shear[I-1]);
    DrawLine(X,0,X,Shear[I-1]);
    Last := I;
    II := No_Bear-1;
END;

IF Dist_Load[No_Load] > Dist_Bear[No_Bear] THEN
BEGIN
X := Dist_Load[No_Load];
I := Round(Dist_Load[No_Load]/DX);
DrawLine( (I-1)*DX,Shear[I-1],X,Shear[I-1]);
DrawLine(X,0,X,Shear[I-1]);
Last := I;
JJ := No_Load-1;
END;

FOR I := J+1 TO Last-1 DO
BEGIN
  IF( I = Round(Dist_Bear[K]/DX) ) AND ( K <= II ) THEN
  BEGIN
    DrawLine( (I-1)*DX,Shear[I-1],I*DX,Shear[I-1] );
    DrawLine( I*DX-0.001,Shear[I-1],I*DX+0.001,Shear[I+1] );
    DrawLine( I*DX,Shear[I+1],(I+1)*DX,Shear[I+1] );
    Temp1 := (I+1)*DX;
    Temp2 := Shear[I+1];
    K := K + 1;
    GOTO 1;
  END;
  IF( I = Round(Dist_Load[L]/DX) ) AND ( L <= JJ ) THEN
  BEGIN
    DrawLine( (I-1)*DX,Shear[I-1],I*DX,Shear[I-1] );
    DrawLine( I*DX-0.001,Shear[I-1],I*DX+0.001,Shear[I+1] );
    DrawLine( I*DX,Shear[I+1],(I+1)*DX,Shear[I+1] );
    Temp1 := (I+1)*DX;
    Temp2 := Shear[I+1];
    L := L + 1;
    GOTO 1;
  END;
  DrawLine(Temp1,Temp2,I*DX,Shear[I]);
  Temp1 := I*DX;
  Temp2 := Shear[I];
1:END;}(for)

GotoXY( 78,10); Write('S');
GotoXY( 78,11); Write('H');
GotoXY( 78,12); Write('E');
GotoXY( 78,13); Write('A');
GotoXY( 78,14); Write('R');

IF Flag = 1 THEN
BEGIN
  SelectScreen(1);
  HardCopy( false,1 );
END; {if}

Repeat
Read( Kbd, ch )
Until ch = ' ';

PROCEDURE Mom_Dia;

VAR
  ch : Char;
  I, J, IX, IY, Flag : Integer;
  X, Y, XX, Sfx, Sy, Temp1, Temp2 : Real;
  Tick, Mark, Bottom, Top, Range : Real;

BEGIN
  ClearScreen;
  GotoXY(1,1);
  Writeln('MOMENT DIAGRAM');
  Writeln;
  Writeln('DO YOU WANT A HARDCOPY OF THE DIAGRAM Y/N ?');
  Write('ANSWER >> '); Read( ch ); ch:= UpCase( ch );
  IF ch = 'Y' THEN Flag:= 1 ELSE Flag:= 0;
  ClearScreen;
  Bottom:= Abs(0.125*(Max_Y-Min_Y));
  Top:= Abs(0.125*(Max_Y-Min_Y));
  DefineWorld(3,-1.1, Min_Y-Bottom, LShaft+1.2, Max_Y+Top);
  SelectWorld(3);
  SelectWindow(1);
  DrawSquare(-1.1, Min_Y-Bottom, LShaft+1.2, Max_Y+Top, false);
  SetLineStyle(1);
  DrawLine(0, Min_Y, LShaft, Min_Y);
  DrawLine(0, Min_Y, 0, Max_Y);
  DrawLine(0,0, LShaft, 0);
  Tick:= LShaft/10;
  Range:= LShaft+2.3;
  Sfx:= 80/Range;
  XX:= Abs(0.25*Bottom);
  SetLineStyle(0);
  FOR I:= 1 TO 10 DO
    BEGIN
      Mark:= Tick*I;
      DrawLine( Mark, Min_Y, Mark, Min_Y+XX );
      X:= 1 + Tick*I;
      IX:= Round( Sfx*X );
      GotoXY( IX, 24 ); Write( Mark:4:1 );
    END; {for}
  Tick:= Round( (Max_Y-Min_Y)/10 );
FOR I:= 1 TO 10 DO
BEGIN
Mark:= Min_Y + Tick*I;
DrawLine( 0, Mark, 0.02*LShaft, Mark );
END; {for}

Range:= (Max_Y + Top) - (Min_Y - Bottom);
Sfy:= 25/Range;
Bottom:= Bottom - 0.125*Bottom;

FOR I:= 1 TO 10 DO
BEGIN
Y:= Bottom + I*Tick;
IY:= 25 - Round( Y*Sfy );
Mark:= Min_Y + I*Tick;
GotoXY( 1,IY ); Write( Mark:4:1 );
END;

Temp1:= 0;
Temp2:= 0;
FOR I:= 0 TO MX-1 DO
BEGIN
DrawLine( Temp1, Temp2, I*DX, Moment[I] );
Temp1:= I*DX;
Temp2:= Moment[I];
END; {for}

GotoXY( 78,9); Write('M');
GotoXY( 78,10); Write('O');
GotoXY( 78,11); Write('M');
GotoXY( 78,12); Write('E');
GotoXY( 78,13); Write('N');
GotoXY( 78,14); Write('T');

IF Flag = 1 THEN
BEGIN
SelectScreen(1);
HardCopy( false,1 );
END; {if}

Repeat
Read( Kbd, ch )
Until ch = ' ';
LeaveGraphic;
END; {procedure Mom_Dia}

PROCEDURE Bear_Select;

LABEL
1;

VAR
ans : Char;
I,J : Integer;
Forc_E,Forc_E1,Forc_E2,Forc_Max,Part_1,Part_2 : Real;
A,Power_1,Power_2,Exponent : Real;

BEGIN
Forc_E:= 0.0;
Forc_Max:= 0.0;
FOR I:= 1 TO No_Bear DO
BEGIN
    J:= Round( Dist_Bear[I]/DX );
    IF Bear_Type[I] = 0 THEN A:= 10/3;
    IF Bear_Type[I] = 1 THEN A:= 3;
    IF I = 1 THEN
        BEGIN
            Forc_E1:= V*Abs(Shear[J]);
            Forc_E2:= 0.5*V*Abs(Shear[J]) + 1.4*Abs(Axial_Thr[1])
        END; {i f}
    IF I = No_Bear THEN
        BEGIN
            Forc_E1:= V*Abs(Shear[J]);
            Forc_E2:= 0.5*V*Abs(Shear[J]) + 1.4*Abs(Axial_Thr[2])
        END; {i f}
    IF ( I <> 1 ) AND ( I <> No_Bear ) THEN
        BEGIN
            Forc_E1:= V*Abs(Shear[J]);
            Forc_E2:= 0.5*V*Abs(Shear[J])
        END; {i f}
IF Forc_E1 > Forc_E2 THEN Forc_E:= Forc_E1 Else Forc_E:= Forc_E2;

    IF Forc_E > Forc_Max THEN
        BEGIN
            Forc_Max:= Forc_E;
            Exponent:= A;
        END; {i f}
END; {for}

Part_1:= (DHours*NSpeed)/6840000.0;
Part_2:= Ln(1/BRel);
Power_1:= 1/A;
Power_2:= 1/(1.17*A);

CRD:= (Forc_Max*Raise(Part_1,Power_1))/Raise(Part_2,Power_2);
Assign(Catalog,Filename);
Reset(Catalog);
WHILE NOT EOF(Catalog) DO
BEGIN
Readln(Catalog,VND,Prt_Num,BORE,OD,WD,FR,DS,DH,CR);
IF ( CR > CRD ) AND ( BORE >= Dia[1] ) THEN GOTO 1;
END; (while)
Writeln;
Writeln('WARNING: NO SUITABLE BEARING FOUND');
Writeln('USING THE LARGEST BEARING IN THE CURRENT
CATALOG');
1:Close( Catalog );
Writeln;
Writeln('BEARING SELECTED : VENDOR: ',VND,' PART
NUM: ',Prt_Num);
Writeln;
Writeln('BEARING LOAD RATING >> ',CR:7:2);
Writeln;
Writeln('CALCULATED LOAD RATING >> ',CRD:7:2);
Writeln;
Writeln('PRESS SPACE BAR TO CONTINUE ...');
Repeat
Read( Kbd,ans )
Until ans = '
';
END; (procedure Bear_Select)

PROCEDURE Thick_Section;

VAR
  ch : Char;
  Top,Middle,WBHalf,Dist,DistA,DistB : Real;
  X,XA,XB,YA,Cnt,Fudge : Integer;
BEGIN
  Top:= BORE*10;
  Middle:= Top - 2.5*BORE;
  WBHalf:= WB/2.0;
  EnterGraphic;
  ResetWorlds;
  ResetWindows;
  DefineWorld(1,-1,0,LShaft+1,BORE*10);
  SelectWorld(1);
  SelectWindow(1);
  DrawSquare(-1, 0, LShaft+1, Top, False);
  DrawSquare(0, Top-2*BORE, LShaft, Top-3*BORE, False);
  SetLineStyle(4);
  FOR I:= 1 TO No_Bear DO
  BEGIN
    DrawLine(DistBear[I], Middle+BoRE, DistBear[I],
    Middle-BORE);
    DrawLine( DistBear[I] - WBHalf, Middle,
    DistBear[I] + WBHalf, Middle);
  END; (for)
  DistA:= Trunc( Dist_Bear[1]/2.0 ) + 1;
DistB := (LShaft - Dist_Bear[No_Bear]) / 2.0 + Dist_Bear[No_Bear] + 1;
XA := Round((80*DistA) / (LShaft+2));
XB := Round((80*DistB) / (LShaft+2));
YA := 25 - Round(25*(Middle - BORE) / (10*BORE));

IF Dist_Bear[1] = 0 THEN Fudge := 0 ELSE Fudge := 1;
IF Dist_Bear[1] = 0 THEN Cnt := 1 ELSE Cnt := 2;

IF Dist_Bear[1] <> 0 THEN
BEGIN
GotoXY(XA,YA); Write('1 ');
END;
IF LShaft - Dist_Bear[No_Bear] <> 0 THEN
BEGIN
GotoXY(XB,YA); Write(No_Bear+Fudge);
END;

FOR I := 2 TO No_Bear DO
BEGIN
Dist := (Dist_Bear[I] - Dist_Bear[I-1]) / 2.0 + Dist_Bear[I-1] + 1;
X := Round((80*Dist) / (LShaft+2));
GotoXY(X,YA); Write(Cnt);
Cnt := Cnt + 1;
END; {for}

FOR I := 1 TO No_Bear DO
BEGIN
X := Round(80*(Dist_Bear[I] + 1) / (LShaft + 2));
YA := 25 - Round(25*(Middle + BORE) / (10*BORE));
GotoXY(X,YA); Write('B ');
END; {for}

IF No_Grv <> 0 THEN
BEGIN
SetLineStyle(4);
FOR I := 1 TO No_Grv DO
BEGIN
DrawLine(Grv_Loc[I],Middle+BORE,Grv_Loc[I],Middle-BORE/2);
X := Round(80*(Grv_Loc[I] + 1) / (LShaft + 2)) - 1;
YA := 25 - Round(25*(Middle+BORE) / (10*BORE));
GotoXY(X,YA); Write('GRV');
END; {for}
END; {if}

IF No_Key <> 0 THEN
BEGIN
FOR I := 1 TO No_Key DO
BEGIN
X := Round(80*(Key_Loc[I] + 1) / (LShaft + 2)) - 1;
YA := 25 - Round(25*(Middle+BORE/2) / (10*BORE));
GotoXY( X,YA ); Write('KEY');
END; {for}
END; {if}
IF No_Hole <> 0 THEN
BEGIN
FOR I:= 1 TO No_Hole DO
BEGIN
  DrawLine(Hole_Loc[I],Middle+BORE,Hole_Loc[I],-
          Middle-BORE/2);
  X:= Round( 80*(Hole_Loc[I]+1)/(LShaft+2) );
  YA:= 25 - Round(25*(Middle+BORE)/(10*BORE));
  GotoXY( X,YA ); Write('H');
END; {for}
END; {if}

FOR I:= 1 TO No_Load DO
BEGIN
  X:= Round( 80*(Dist_Load[I]+1)/(LShaft+2) ) - 1;
  YA:= 25 - Round(25*Middle/(10*BORE));
  GotoXY( X,YA ); Write('HUB');
END; {for}

GotoXY( 2,13 ); Write('B -- DENOTES BEARING LOCATIONS');
GotoXY( 2,14 ); Write('H -- DENOTES HOLE LOCATIONS');
GotoXY( 2,15 ); Write('GRV -- DENOTES GROOVE LOCATIONS');
GotoXY( 2,17 ); Write('ENTER THE NUMBER OF THE THICK
SECTION >> ');
Sect_Th := Readln;
GotoXY( 2,19 ); Write('ENTER THE RADIUS OF THE FILLETS >>
');
Fillet := Readln;

IF (Dist_Bear[1] <> 0) AND (Sect_Th = 1) THEN Str_Raise:= 1;
IF (Dist_Bear[1] <> 0) AND (Sect_Th > No_Bear) THEN
  Str_Raise:= 1;
IF (Dist_Bear[1] = 0 ) AND (Sect_Th = No_Bear) THEN
  Str_Raise:= 1;

Step1:= 0;
Step2:= 0;
IF (Str_Raise = 2) AND (Dist_Bear[1] = 0) THEN
BEGIN
  Step1:= Dist_Bear[Sect_Th] + WB/2;
  Step2:= Dist_Bear[Sect_Th+1] - WB/2;
END;
IF (Str_Raise = 2) AND (Dist_Bear[1] <> 0) THEN
BEGIN
  Step1:= Dist_Bear[Sect_Th-1] + WB/2;
  Step2:= Dist_Bear[Sect_Th] - WB/2;
END;
PROCEDURE Steps1;

LABEL 1;

VAR
  I : Integer;
  Sum : Real;

BEGIN
  Assign( Catalog, Filename );
  Reset( Catalog );
  WHILE NOT EOF( Catalog ) DO
    BEGIN
      Readln( Catalog, VND, Prt_Num, BORE, OD, WB, FR, DS, DH, CR );
      IF ( CR > CRD ) AND ( BORE >= Dia[1] + 0.125 ) THEN
        GOTO 1;
    END; {while}
  1: Close( Catalog );

  NumH := 0;
  Sum := 0;
  IF Sect_Th = 1 THEN
    BEGIN
      Step1 := Dist_Bear[1] - WB/2.0;
      FOR I := 1 TO No_Load DO
        BEGIN
          IF Dist_Load[I] < Step1 THEN NumH := NumH + 1;
        END; {for}
    END; {if}

  IF Sect_Th >= No_Bear THEN
    BEGIN
      Step1 := Dist_Bear[No_Bear] + WB/2.0;
      FOR I := 1 TO No_Load DO
        BEGIN
          IF Dist_Load[I] > Step1 THEN NumH := NumH + 1;
        END; {for}
    END; {if}

  IF NumH > 1 THEN
    BEGIN
      Writeln;
      Writeln( Bl:10, '**** THIS CASE EXCEEDS THE PROGRAM CAPABILITIES ****' );
      Writeln;
    END; {if}
Writeln(B1:24,'PROGRAM TERMINATED');
Halt;
END; {if}

IF NumH = 0 THEN
BEGIN
  Writeln;
  Writeln('THERE ARE NO HUBS WITHIN THIS SECTION -- NO STEPS REQUIRED ?');
END;

IF ( NumH = 1 ) AND ( Sect_Th = 1 ) THEN
BEGIN
  Sec_Len[1]:= Dist_Load[1] - Width_Hub[1]/2.0;
  Dia[1]:= DS;
  Sec_Len[2]:= Width_Hub[1];
  Dia[2]:= DS + 0.125;
  Sec_Len[3]:= Step1/10.0;
  Dia[3]:= DS + 0.25;
  Sec_Len[4]:= Step1 - Sum;
  Dia[4]:= DS;
  Sec_Len[5]:= WB;
  Dia[5]:= BORE;
  Sec_Len[6]:= LShaft - ( Step1 + WB );
  Dia[6]:= BORE - 0.125;
  No_SecChDia:= 6;
END; {if}

IF ( NumH = 1 ) AND ( Sect_Th >= No_Bear ) THEN
BEGIN
  Sec_Len[1]:= Step1 - WB;
  Dia[1]:= BORE - 0.125;
  Sec_Len[2]:= WB;
  Dia[2]:= BORE;
  X:= ( LShaft - Step1 )/10.0;
  Sec_Len[3]:= ( Dist_Load[No_Load] - Width_Hub[No_Load]/2 ) - X - Step1;
  Dia[3]:= DS;
  Sec_Len[4]:= ( LShaft - Step1 )/10.0;
  Dia[4]:= DS + 0.25;
  Sec_Len[5]:= Width_Hub[No_Load];
  Dia[5]:= DS + 0.125;
  Sec_Len[6]:= LShaft - ( Dist_Load[No_load] + Width_Hub[No_Load]/2.0 );
  Dia[6]:= DS;
  No_SecChDia:= 6;
END; {if}
END; {procedure steps1}

PROCEDURE Steps2;
VAR
  I : Integer;
  Sum : Real;
BEGIN
  NumH:= 0;
  Sum:=0;
  WhereToStart:= 0;
  FOR I:= 1 TO No_Load DO
  BEGIN
    IF ( Dist_Load[I] > Step1 ) AND ( Dist_Load[I] < Step2 ) THEN
      BEGIN
        NumH := NumH + 1;
        Point:= I;
      END
    END
  IF NumH = 0 THEN
    BEGIN
      Writeln;
      Writeln('THERE ARE NO HUBS WITHIN THIS SECTION -- NO STEPS REQUIRED ?');
    END
  IF NumH > 3 THEN
    BEGIN
      Writeln;
      Writeln(B1:10,'**** THIS CASE EXCEEDS THE PROGRAM CAPABILITIES ****');
      Writeln;
      Writeln(B1:24,'PROGRAM TERMINATED');
      Halt;
    END
  IF NumH = 1 THEN
    BEGIN
      WhereToStart:= Point;
      K:= WhereToStart;
      Sec_Len[1]:= Step1 - WB;
      Dia[1]:= BORE - 0.125;
      Sec_Len[2]:= WB;
      Dia[2]:= BORE;
      Sec_Len[3]:= ( Dist_Load[K] - Width_Hub[K]/2 ) - Step1;
      Dia[3]:= DS;
      Sec_Len[4]:= Width_Hub[K];
      Dia[4]:= DS + 0.125;
      Sec_Len[5]:= (Step2 - Step1)/10;
      Dia[5]:= DS + 0.25;
END
Sec_Len[6]:= Step2 - Sum;
Dia[6]:= DS;
Sec_Len[7]:= WB;
Dia[7]:= BORE;
Sec_Len[8]:= LShaft - ( Step2 + WB );
Dia[8]:= BORE - 0.125;
No_SecChDia:= 8;
END; {if}

IF NumH = 2 THEN
BEGIN
   WhereToStart:= (Point - NumH) + 1;
   K:= WhereToStart;
   Sec_Len[1]:= Step1 - WB;
   Dia[1]:= BORE - 0.125;
   Sec_Len[2]:= WB;
   Dia[2]:= BORE;
   Sec_Len[3]:= (Dist_Load[K] - Width_Hub[K]/2) - Step1;
   Dia[3]:= DS;
   Sec_Len[4]:= Width_Hub[K];
   Dia[4]:= DS + 0.125;
   Sec_Len[5]:= (Dist_Load[K+1] - Width_Hub[K+1]/2) - Sum;
   Dia[5]:= DS + 0.25;
   Sec_Len[6]:= Width_Hub[K+1];
   Dia[6]:= DS + 0.125;
   Sum:= Sum + Sec_Len[5] + Sec_Len[6];
   Sec_Len[7]:= Step2 - Sum;
   Dia[7]:= DS;
   Sec_Len[8]:= WB;
   Dia[8]:= BORE;
   Sec_Len[9]:= LShaft - ( Step2 + WB );
   Dia[9]:= BORE - 0.125;
   No_SecChDia:= 9;
END; {if}

IF NumH = 3 THEN
BEGIN
   WhereToStart:= (Point - NumH) + 1;
   K:= WhereToStart;
   Sec_Len[1]:= Step1 - WB;
   Dia[1]:= BORE - 0.125;
   Sec_Len[2]:= WB;
   Dia[2]:= BORE;
   Sec_Len[3]:= (Dist_Load[K] - Width_Hub[K]/2) - Step1;
   Dia[3]:= DS;
   Sec_Len[4]:= Width_Hub[K];
   Dia[4]:= DS + 0.125;
DIA[5] := DS + 0.25;
SEC_LEN[6] := WIDTH_HUB[K+1];
DIA[6] := DS + 0.375;
DIA[7] := DS + 0.5;
SEC_LEN[8] := WIDTH_HUB[K+2];
DIA[8] := DS + 0.25;
DIA[9] := DS;
SEC_LEN[10] := WB;
DIA[10] := BORE;
SEC_LEN[11] := LSHAFT - (STEP2 + WB);
NO_SECCHDIA := 11;
END; {if}
END; {procedure steps2}

PROCEDURE Min_MaxMEI;
VAR
I : Integer;
BEGIN
Temp1 := 100000.0;
Temp2 := -100000.0;
FOR I := 0 TO MX-1 DO
BEGIN
IF MOM_EI[I] >= Temp2 THEN
BEGIN
Max_Y := MOM_EI[I];
Temp2 := Max_Y;
END;
IF MOM_EI[I] <= Temp1 THEN
BEGIN
Min_Y := MOM_EI[I];
Temp1 := Min_Y;
END;
END; {for}
END; {procedure Min_MaxMEI}

PROCEDURE MomEI_Dia;
VAR
ch : Char;
I,J,IX,IY,Flag : Integer;
X,Y,XX,Sfx,Sfy,Templ,Temp2 : Real;
Tick,Mark,Bottom,Top,Range : Real;

BEGIN
EnterGraphic;
  GotoXY(1,1);
  Writeln(' [MOMENT*1000]/[E*I] DIAGRAM');
  Writeln;
  Writeln('DO YOU WANT A HARDCOPY OF THE DIAGRAM Y/N ?');
  Write('ANSWER >> ');
  Read( ch ); ch:= UpCase( ch );
  IF ch = 'Y' THEN Flag:= 1 ELSE Flag:= 0;
  ClearScreen;
  Bottom:= Abs(0.125*(Max_Y-Min_Y));
  Top:= Abs(0.125*(Max_Y-Min_Y));

DefineWorld(2,-1.1,Min_Y-Bottom,LShaft+1.2,Max_Y+Top);
  SelectWorld(2);
  SelectWindow(1);

DrawSquare(-1.1,Min_Y-Bottom,LShaft+1.2,Max_Y+Top,false);

SetLineStyle(1);
  DrawLine(0,Min_Y,LShaft,Min_Y);
  DrawLine(0,Min_Y,0,Max_Y);
  DrawLine(0,0,LShaft,0);

Tick:= LShaft/10;
Range:= LShaft+2.3;
Sfx:= 80/Range;
XX:= Abs(0.25*Bottom);
SetLineStyle(0);

FOR I:=1 TO 10 DO
BEGIN
  Mark:= Tick*I;
  DrawLine( Mark, Min_Y, Mark, Min_Y+XX );
  X:= 1 + Tick*I;
  IX:= Round( Sfx*X );
  GotoXY( IX,24 ); Write( Mark:4:1 );
END; {for}

Tick:= (Max_Y-Min_Y)/10;

FOR I:= 1 TO 10 DO
BEGIN
  Mark:= Min_Y + Tick*I;
  DrawLine( 0, Mark, 0.02*LShaft, Mark );
END; {for}
Range:= (Max_Y + Top) - (Min_Y - Bottom);
Sfy:= 25/Range;
Bottom:= Bottom - 0.125*Bottom;

FOR I:= 1 TO 10 DO
BEGIN
  Y:= Bottom + I*Tick;
  IY:= 25 - Round( Y*Sfy );
  Mark:= Min_Y + I*Tick;
  GotoXY( 1,IY ); Write( Mark:6:3 );
END;

Temp1:= 0;
Temp2:= 0;
FOR I:= 0 TO MX-1 DO
BEGIN
  DrawLine( Temp1, Temp2, I*DX, Mom_EI[I] );
  Temp1:= I*DX;
  Temp2:= Mom_EI[I];
END; {for}

GotoXY( 73,9); Write('M*1000');
GotoXY( 75,10); Write('----');
GotoXY( 76,11); Write('EI');

IF Flag = 1 THEN
BEGIN
  SelectScreen(1);
  HardCopy( false,1 );
END; {if}

Repeat
  Read( Kbd, ch )
Until ch = ' ';
LeaveGraphic;
END; {procedure MomEl_Dia}

BEGIN
FOR I:= 1 TO 10 DO
BEGIN
  Bear_Pos[I]:= 0;
  Dist_Bear[I]:= 0.0;
END;
ClrScr;
Writeln;
Writeln('**** MATERIAL INFORMATION ****'); Writeln;
REPEAT
  Writeln('MATERIAL 1. CARBON STEEL  2. ALUMINUM');
  Write('CHOOSE A NUMBER >> ');
  Readln(Choice);
  CASE Choice OF
'1': Mod_Elast := 3.0E+07;
'2': Mod_Elast := 1.0E+07;
ELSE Writeln('ERROR: INVALID SELECTION, PLEASE TRY AGAIN.');
END; {case}
UNTIL ( Choice = '1' ) OR ( Choice = '2' );
Writeln;
Write('ENTER THE YIELD STRENGTH (psi) >> ');
Readln(S_Ys);
Writeln;
Write('ENTER THE ULTIMATE TENSILE STRENGTH (psi) >> ');
Readln(S_Ult);
ClrScr;
Writeln; Writeln('**** SHAFT LENGTH & BEARING INFORMATION ****');
Writeln; Write('ENTER THE LENGTH (in.) OF THE SHAFT >> ');
Readln(LShaft);
Writeln;
Repeat
Write('ENTER THE NUMBER OF BEARINGS >> ');
Readln(No_Bear);
IF No_Bear <= 1 THEN Writeln('ONE BEARING YIELDS AN UNSTABLE PROBLEM -- IMPOSSIBLE SOLUTION');
Until No_Bear > 1;
FOR I:= 1 TO No_Bear DO
BEGIN
    Writeln;
    Write('ENTER THE DISTANCE(in.) TO BEARING(',I,') >> ');
    Readln( Dist_Bear[I] );
END; {for}

Design_Area;

FOR I:= 1 TO 11 DO
BEGIN
    Dia[I]:= 0.0;
    MomI[I]:= 0.0;
    Sec_Len[I]:= 0.0;
END;

FOR I:= 1 TO 10 DO
BEGIN
    Grv_Loc[I]:= 0.0;
    Key_Loc[I]:= 0.0;
    Hole_Loc[I]:= 0.0;
    Dist_Load[I]:= 0.0;
    Load[I]:= 0.0;
    Load_Loc[I]:= 0;
END;
FOR I:= 0 TO MX-1 DO
BEGIN
    Torque[I]:= 0.0;
    D[I]:= 0.0;
    Moment[I]:= 0.0;
    Shear[I]:= 0.0;
    Mom_EI[I]:= 0.0;
END; {if}

X:= LShaft/20;
Y:= Round( X*16);
Dia[1]:= Y/16;
MomI[1] := (PI*Dia[1]*Dia[1]*Dia[1]*Dia[1])/64;

No_Seg := 50;
DX := LShaft/No_Seg;
Delta_Cong := DX*DX*DX;

Writeln; Writeln('**** HUB & LOAD INFORMATION ****');
Writeln; Write('ENTER THE NUMBER OF HUBS » '); Readln( No_Load);
FOR I:= 1 TO No_Load DO
BEGIN
    Writeln;
    Write('ENTER THE DISTANCE (in.) TO HUB(',I,') » '); Readln( Dist_Load[I] );
    Write('ENTER THE WIDTH (in.) OF HUB(',I,') » '); Readln( Width_Hub[I] );
    Write('ENTER THE CONC.LOAD(lbs) ON HUB(',I,') » '); Readln( Load[I] );
    CLoad[I]:= Load[I];
    Load_Loc[I]:= Round( Dist_Load[I]/DX ) + 1;
    CLoad_Loc[I]:= Load_Loc[I];
    Write('ENTER THE TORQUE(in-lbs) ON HUB(',I,') » '); Readln( Torq[I] );
    Write('ENTER THE THRUST(lbs) ON HUB(',I,') » '); Readln( Thrust[I] );
END; { for }

TFx:= 0.0;
FOR I:= 1 TO No_Load DO
    TFx:= TFx + Thrust[I];

Axial_Thr[1]:= 0.0;
Axial_Thr[2]:= 0.0;
IF TFx > 0 THEN Axial_Thr[2]:= -TFx ELSE Axial_Thr[1]:= -TFx;
IF TFx > 0 THEN Compr_End:= Round(Dist_Bear[No_Bear]/DX) ELSE Compr_End:= Round(Dist_Bear[1]/DX);
IF No_Load > 1 THEN TOR; { FINDS THE TORQUE DISTRIBUTION IN THE SHAFT }

IF No_Bear > 0 THEN
BEGIN
  FOR I:= 1 TO No_Bear DO {find bearing matrix position}
    Bear_Pos[I] := Round ( Dist_Bear[I]/DX );
END; (if)

MatSize := Trunc( LShaft/DX - (No_Bear - 1) );

IF Bear_Pos[1] = 1 THEN MatSize := MatSize-1;
IF Bear_Pos[No_Bear] = Trunc( (LShaft-DX)/DX THEN
  MatSize := MatSize-1;

Writeln; Writeln('PERFORMING CALCULATIONS --- PLEASE WAIT.');
Writeln; Writeln(' STARTING DIAMETER = ',Dia[1]:6:4);
Writeln;

GetMem( Stiff, Sizeof( Matrix ) );
K:= 0;
1: In_Matrix;

Over_Write;

Inertia;

Invert;

Temp:= -1.0;
FOR I:= 1 TO MatSize DO
  IF Abs( Stiff[I,MatSize+1] ) > Temp THEN Temp:= Abs( Stiff[I,MatSize+1] );
IF K < 6 THEN
BEGIN
  Writeln(' MAX. DEFLECTION = ',Temp:7:5);
  IF Temp > 0.01 THEN
    BEGIN
      Writeln('Diameter 1 = ',Dia[1]:5:4);
      IF Temp > 0.09 THEN Dia[1]:= Dia[1] + 0.5;
      IF ( Temp <= 0.09 ) AND ( Temp > 0.04 ) THEN Dia[1]:= Dia[1] + 0.1875;
      IF ( Temp <= 0.04 ) AND ( Temp > 0.014 ) THEN Dia[1]:= Dia[1] + 0.125;
      IF ( Temp <= 0.014 ) AND Temp > 0.01 THEN Dia[1]:= Dia[1] + 0.0625;
      Writeln('Increase Diameter = ',Dia[1]:5:4);
      MomI[1]:= (PI*Dia[1]*Dia[1]*Dia[1]*Dia[1])/64;
      FOR J:= 1 TO No_Load DO
        BEGIN

Load[J]:= CLoad[J];
Load_Loc[J]:= CLoad_Loc[J];
END;

WRITEln('DEFLECTION OF SHAFT EXCEEDS 0.01');
K:= K + 1;
WRITEln; WRITEln(B1:22,'ITERATION ',K);
GOTO 1;
END; {if}
END; {if}
IF K = 6 THEN WRITEln('DEFLECTIONS MAY EXCEED 0.01 BUT CONTINUING PROGRAM EXECUTION');

COPY_Deflect;

FREEMem( Stiff, sizeof( Matrix ) );

Beep;

VShear;

TMoment;

Max_MinSH;

SH_Dia;

Min_MaxMom;

Mom_Dia;

Repeat
WRITEln;
WRITEln('DO YOU WANT A HARDCOPY OF THE LOAD AND STRESS DISTRIBUT. ON SHAFT ? ');
WRITE('ANSWER Y OR N >> ');
READln( ans ); ans:= UpCase( ans );
IF ans = 'Y' THEN HDCopy;
UNTIL ( ans = 'Y' ) OR ( ans = 'N' );

CLRScr;
WRITEln; WRITEln('*** BEARING SELECTION ***');

REPEAT
WRITEln;
WRITE('WHICH RACE ROTATES (I)NNER OR (O)UTER >> ');
READln( Choice ); Choice:= UpCase( Choice );
CASE Choice OF
'I' : V:= 1.0;
'O' : V:= 1.2;
ELSE WRITEln('ERROR: INVALID SELECTION, PLEASE TRY AGAIN');
END
UNTIL ( Choice = 'I' ) OR ( Choice = 'O' );
111

Writeln;
Write('ENTER THE SPEED OF ROTATION (RPM) >> '); Readln( NSpeed );
Write('ENTER THE DESIGN LIFE (HOURS) >> '); Readln( DHours );
Write('ENTER THE BEARING RELIABILITY >> '); Readln( BRel );
Writeln;
Write('ENTER THE BEARING CATALOG FILENAME >> '); Readln( Filename );
FOR I:= 1 TO Length( Filename ) DO Filename[I]:= UpCase( Filename[I] );
Filename:= Concat( Filename,Ftype );

FOR I:= 1 TO No_Bear DO BEGIN
  Writeln;
  REPEAT
    Write('IS BEARING(',I,') (R)OLLER OR (B)ALL ? >> '); Readln( Choice );
    Choice:= UpCase( Choice );
    CASE Choice OF
      'R' : Bear_Type[I]:= 0;
      'B' : Bear_Type[I]:= 1;
      ELSE Writeln('ERROR: INVALID SELECTION, PLEASE TRY AGAIN');
    END
  UNTIL ( Choice = 'R' ) OR ( Choice = 'B' );
END; {for}

Bear_Select;
ClrScr;
Writeln;
Writeln('THE FOLLOWING QUESTIONS ARE FOR THE TENATIVE LOCATIONS OF');
Writeln(' HOLES, GROOVES, AND KEWAYS');
Writeln;
Write('ENTER THE NUMBER OF HOLES >> '); Readln( No_Hole );
IF No_Hole <> 0 THEN BEGIN
  FOR I:= 1 TO No_Hole DO BEGIN
    Writeln;
    Write('ENTER THE LOCATION OF HOLE(',I,') >> ');
    Readln( Hole_Loc[I] );
  END; {for}
END; {if}
Writeln;
Write('ENTER THE NUMBER OF KEYWAYS >> '); Readln( No_Key );
IF No_Key <> 0 THEN
BEGIN
  FOR I:= 1 TO No_Key DO
  BEGIN
    Writeln;
    Write('ENTER THE LOCATION OF KEYWAY(',I,') » I);
    Readln( Key_Loc[I] );
  END; {for}
END; {if}
Writeln;
Write('ENTER THE NUMBER OF GROOVES » I);
Readln( No_Grv );
IF No_Grv <> 0 THEN
BEGIN
  FOR I:= 1 TO No_Grv DO
  BEGIN
    Writeln;
    Write('ENTER THE LOCATION OF GROOVE(',I,') » I);
    Readln( Grv_Loc[I] );
  END; {for}
END; {if}
Writeln;
Writeln('PRESS THE SPACE BAR TO CONTINUE ...');

Repeat
  Read( Kbd, ans )
Until ans = ' ';

Thick_Section;

IF Str_Raise = 1 THEN Steps1;
IF Str_Raise = 2 THEN Steps2;

IF NumH <> 0 THEN
BEGIN
  FOR I:= 1 TO No_SecChDia DO
    MomI[I] := (PI*Dia[I]*Dia[I]*Dia[I]*Dia[I])/64;
END; {if}

IF NumH <> 0 THEN
BEGIN
  J := 1;
  Sum := Sec.Len[J];
  FOR I:= 0 TO MX-1 DO
  BEGIN
    IF J < No_SecChDia THEN
      BEGIN
        IF (I*DX) > Sum THEN
          BEGIN
            J := J + 1;
            Sum := Sum + Sec.Len[J];
          END;
      END;
  END;
END;
END; {if}
  END; {if}
  Mom_EI[I] := 1000*Moment[I]/(Mod_Elast*MomI[J]);
END; {for}
END; {if}

Min_MaxMEI;

MomEI_Dia;

Change_Prop := 0;
Assign(ChainFile,'SubProgm.chn');
Chain(ChainFile);
END. {program}
PROGRAM SubProgm;
{$I typedef.sys}
{$I graphix.sys}
{$I kernel.sys}

LABEL
  1,2;

CONST
  PI = 3.141592;
  Bl = ' ';
  MX = 51;
  MAX = 52;
  Ftype = '.DAT';
  Max_Char = 6;

TYPE
  Real2 = ARRAY[1..2] OF Real;
  Real11= ARRAY[1..11] OF Real;
  Vals = ARRAY[0..MX] OF Real;
  Copy = ARRAY[0..MX] OF Real;
  NValues = ARRAY[1..10] OF Real;
  IValues = ARRAY[1..10] OF Integer;
  Strings = Strings[Max_Char];
  Matrix = ARRAY[1..MX,1..MAX] OF Real;

VAR
  ans,Opt,Choice : Char;
  Catalog : Text;
  Filename: String[12];
  VND,Prt_Num : Strings;
  Change.Prop : Integer;
  S_E,F_Safe,Cal_D1,Cal_D2,Tes_D1,Tes_D2 : Real;
  Ktbgr,Ktbh,Ktbst,KA,KB,KC,KE,Kf,M_Mom,M_Torq,A_Torq,A_Mom,-
  Max_S1,Max_S2 : Real;
  No_Grv,No_Key,No_Hole,NumH,Point,WhereToStart,Flag :
    Integer;
  BORE,OD,WD,FR,DS,DH,CR,Fillet,APower,D_Ratio,R_Ratio,-
  Keytb,Rad,q : Real;

  X,Y,R,Temp,Temp1,Temp2,Max_Y,Min_Y,CRD,V,BRel,NSpeed,Step1-,
  Step2 : Real;
  Mod_Elast,S_Ys,S_Ult,LShaft,DX,Delta_Conc,TFx,DHours,-
  Sum_Torq,Sum : Real;

  Start,Finish,No_SecChDia,No_Load,Set_Up,Sect_Th,Str_Raise,-
  Compr_End : Integer;

  Last,II, JJ,Tick,Mark,MatSize,I,J,K,L,Cnt,No_Bear,No_Seg,MS :
    Integer;
  Bear_Pos : IValues;
FUNCTION Sign( Number : Real ):Real;
BEGIN
  IF Number = 0.0
  THEN Sign:= 1
  ELSE Sign:= Abs( Number )/Number
END; {function sign}

FUNCTION Raise( Number,Power : Real ):Real;
BEGIN
  IF Number = 0.0
  THEN IF Power = 0.0
      THEN Raise:= 1.0
      ELSE Raise:= 0.0
  ELSE Raise:= Sign( Number)*Exp( Power*Ln( Abs( Number )))
END; {function raise}

PROCEDURE Mod_FacKE;

Label
  1,2;

VAR
  I,J : Integer;
BEGIN
Sum := 0.0;
J := 1;
IF No_Grv <> 0 THEN
BEGIN
FOR I := 1 TO No_SecChDia DO
BEGIN
Sum := Sum + Sec.Len[I];
IF J <= No_Grv THEN
BEGIN
IF Grv_Loc[J] <= Sum THEN
(*1:BEGIN*)
D_Ratio := Dia[I]/Grv_ID[J];
IF (D_Ratio > 1.0) AND (D_Ratio <= 1.14) THEN APower :=
-0.1738*Raise(D_Ratio, 3.72);
IF (D_Ratio > 1.14) AND (D_Ratio <= 1.5) THEN APower :=
-0.1963 - 0.0764*D_Ratio;
IF D_Ratio > 1.5 THEN APower := -0.311;
R_Ratio := Grv_Rad[J]/Grv_ID[J];
Rad := Grv_Rad[J];
Ktbgr := 0.9236*Raise(R_Ratio, APower);
IF Mod_Elast = 3.0E+07 THEN
BEGIN
IF S_Ult >= 1.6E+05 THEN q := 1.06853 +
0.047466*Ln(Rad);
IF (S_Ult < 1.6E+05) AND (S_Ult >=
1.2E+05) THEN q := 1.0533 + 0.06667*Ln(Rad);
IF (S_Ult < 1.2E+05) AND (S_Ult >=
8.0E+04) THEN q := 1.0422 + 0.09374*Ln(Rad);
IF S_Ult < 8.0E+04 THEN q := 0.99758 +
0.11214*Ln(Rad);
END;
IF Mod_Elast = 1.0E+07 THEN q := 1.18705 +
0.198248*Ln(Rad);
Kf := 1 + q*(Ktbgr - 1);
IF Kf > Temp1 THEN Temp1 := Kf;
J := J + 1;
END; (if)
IF (J <= No_Grv) AND (Grv_Loc[J] <= Sum) THEN
GOTO 1;
ENDIF
END; (for)
END; (if)
END; (if)

Sum := 0.0;
J := 1;
IF No_Hole <> 0 THEN
BEGIN
FOR I := 1 TO No_SecChDia DO
BEGIN

Sum:= Sum + Sec_Len[I];
IF J <= No_Hole THEN
BEGIN
  IF Hole_Loc[J] <= Sum THEN
    2: BEGIN
      D_Ratio:= Hole_D[J]/Dia[I];
      Rad:= Hole_D[J]/2;
      Ktbh:= 1.642*Raise(D_Ratio,-0.1305);
      IF Mod_Elast = 3.0E+07 THEN
        BEGIN
          IF S_Ult >= 1.6E+05 THEN q:= 1.06853 +
            0.027466*Ln(Rad);
          IF (S_Ult < 1.6E+05) AND ( S_Ult >=
            1.2E+05) THEN q:= 1.0533 + 0.06667*Ln(Rad);
          IF (S_Ult < 1.2E+05) AND ( S_Ult >=
            8.0E+04) THEN q:= 1.0422 + 0.09375*Ln(Rad);
          IF S_Ult < 8.0E+04 THEN q:= 0.99758 +
            0.11214*Ln(Rad);
        END;
      IF Mod_Elast = 1.0E+07 THEN q:= 1.18705 +
        0.198248*Ln(Rad);
      Kf:= 1 + q*(Ktbh - 1);
      IF Kf > Temp THEN Temp:= Kf;
      J:= J + 1;
      END; {if}
      IF ( J <= No_Hole ) AND ( Hole_Loc[J] <=
        Sum ) THEN GOTO 2;
    END; {if}
  END; {for}
END; {if}

FOR I:= 1 TO No_SecChDia-1 DO
BEGIN
  D_Ratio:= Dia[I+1]/Dia[I];
  IF D_Ratio < 1.0 THEN D_Ratio:= 1/D_Ratio;
  IF Dia[I+1] < Dia[I] THEN R_Ratio:= Fillet/Dia[I+1]
  ELSE R_Ratio:= Fillet/Dia[I];
  IF (D_Ratio > 1.0) AND (D_Ratio <= 1.11) THEN
    APower:= -0.704*D_Ratio + 0.5327;
  IF (D_Ratio > 1.11) AND (D_Ratio <= 3.0) THEN
    APower:= -0.251*Raise(D_Ratio,0.2169);
  IF D_Ratio > 3.0 THEN APower:= -0.319;
  Ktbst:= 0.921*Raise(R_Ratio,APower);
  IF Mod_Elast = 3.0E+07 THEN
    BEGIN
      IF S_Ult >= 1.6E+05 THEN q:= 1.0685 +
        0.047466*Ln(Fillet);
      IF (S_Ult < 1.6E+05) AND ( S_Ult >= 1.2E+05)
        THEN q:= 1.0533 + 0.06667*Ln(Fillet);
      IF (S_Ult < 1.2E+05) AND ( S_Ult >= 8.0E+04)
        THEN q:= 1.0422 + 0.09374*Ln(Fillet);
      IF S_Ult < 8.0E+04 THEN q:= 0.99758 +
        0.11214*Ln(Fillet);
    END; {if}
"
IF S_Ult < 8.0E+04 THEN q := 0.99758 + 0.11214*Ln(Fillet);
END; (if)
 IF Mod_Elast = 1.0E+07 THEN q := 1.18705 + 0.19824*Ln(Fillet);
 Kf := 1 + q*(Kbst - 1);
 IF Kf > Temp2 THEN Temp2 := Kf;
 END; (for)
END; (procedure Mod_FacKE)

PROCEDURE Mod_FacKE1;
BEGIN
 IF S_Ult >= 1.6E+05 THEN q := 0.895;
 IF (S_Ult < 1.6E+05) AND (S_Ult >= 1.2E+05) THEN q := 0.79;
 IF (S_Ult < 1.2E+05) AND (S_Ult >= 8.0E+04) THEN q := 0.735;
 IF S_Ult < 8.0E+04 THEN q := 0.625;
 Kf := 1 + q*(Kbst - 1);
 IF Kf > Temp2 THEN Temp2 := Kf;
END; (procedure Mod_FacKE1)

PROCEDURE Mod_FacKE2;
BEGIN
 D_Ratio := DS/BORE;
 R_Ratio := FR/BORE;
 IF (D_Ratio > 1.0) AND (D_Ratio <= 1.11) THEN APower := -0.704*D_Ratio + 0.5327;
 IF (D_Ratio > 1.11) AND (D_Ratio <= 3.0) THEN APower := -0.251*Raise(D_Ratio,0.1269);
 IF D_Ratio > 3.0 THEN APower := -0.319;
 Kbst := 0.921*Raise(R_Ratio,APower);
 IF Mod_Elast = 3.0E+07 THEN
 BEGIN
 IF S_Ult > 1.6E+05 THEN q := 1.0685 + 0.047466*Ln(FR);
 IF (S_Ult > 1.2E+05) AND (S_Ult <= 1.6E+05) THEN q := 1.0533 + 0.06667*Ln(FR);
 IF (S_Ult > 8.0E+04) AND (S_Ult <= 1.2E+05) THEN q := 1.0422 + 0.09374*Ln(FR);
 IF S_Ult <= 8.0E+04 THEN q := 0.99758 + 0.11214*Ln(FR);
 END; (if)
 IF Mod_Elast = 1.0E+07 THEN q := 1.18705 + 0.19824*Ln(FR);
 Kf := 1 + q*(Kbst - 1);
 IF Kf > Temp2 THEN Temp2 := Kf;
END; (procedure Mod_FacKE2)
PROCEDURE Mod_FacKB;

VAR
    I, IX, J : Integer;
    Sum, Max : Real;

BEGIN
    J := 1;
    Max := 0.0;
    Sum := Sec_Len[1];
    FOR I := 1 TO MX-1 DO
        BEGIN
            IF NumH <> 0 THEN
                BEGIN
                    IF J < No_SecChDia THEN
                        BEGIN
                            IF I*DX > Sum THEN
                                BEGIN
                                    J := J + 1;
                                    Sum := Sum + Sec_Len[J];
                                END;
                            END;
                        END;
                    END;
                    IF Abs( Mom_EI[I] ) > Max THEN
                        BEGIN
                            Tes_D1 := Dia[J];
                            Max := Abs( Mom_EI[I] );
                            M_Mom := Abs( Moment[I] );
                            A_Torq := Abs( Torque[I] );
                            Max_S1 := (16*A_Torq)/(PI*Dia[J]*Dia[J]*Dia[J]);
                        END;
                    END;
                END;
            END;
            IF Tes_D1 < 0.3 THEN KB := 1;
            IF ( Tes_D1 >= 0.3 ) AND ( Tes_D1 < 2 ) THEN KB := 0.85;
            IF Tes_D1 >= 2 THEN KB := 0.75;
        END;
    END;
END;

PROCEDURE MX_Torq;

VAR
    I, J : Integer;
    Sum, Max, Temp, M_Shear : Real;

BEGIN
    J := 1;
    Max := 0.0;
    Temp := 0.0;
    Sum := Sec_Len[1];
    FOR I := 1 TO MX-1 DO
        BEGIN
            IF Tes_D1 < 0.3 THEN KB := 1;
            IF ( Tes_D1 >= 0.3 ) AND ( Tes_D1 < 2 ) THEN KB := 0.85;
            IF Tes_D1 >= 2 THEN KB := 0.75;
        END;
END;
IF NumH <> 0 THEN
BEGIN
  IF J < No_SecChDia THEN
  BEGIN
    IF I*DX > Sum THEN
    BEGIN
      J := J + 1;
      Sum := Sum + Sec_Len[J];
    END; (if)
  END; (if)
END; (if)

M_Shear := (16*Abs(Torque[I]))/(PI*Dia[J]*Dia[J]*Dia[J]);

IF M_Shear >= Max THEN
BEGIN
  IF M_Shear = Max THEN
  BEGIN
    IF Abs(Moment[I]) > Temp THEN
    BEGIN
      Temp := Abs(Moment[I]);
      A_Mom := Abs(Moment[I]);
      Tes_D2 := Dia[J];
      M_Torq := Abs(Torque[I]);
      Max_S2 := M_Shear;
    END; (if)
  END; (if)
  IF M_Shear > Max THEN
  BEGIN
    Temp := Moment[I];
    Max := M_Shear;
    A_Mom := Abs(Moment[I]);
    Tes_D2 := Dia[J];
    M_Torq := Abs(Torque[I]);
    Max_S2 := M_Shear;
  END;
END; (if)
END; (for)

END; (procedure M_Torq)

PROCEDURE Surface;

VAR
  A0,A1,A2,S : Real;

BEGIN
  S := S_Ult/1000;
  A0 := 1 - 8.5E-05*R - 9.5E-07*R*R;
  A1 := 5.16E-05 - 2.86E-05*R + 4.22E-08*R*R;
  A2 := 1.79E-07 + 5.2E-08*R - 8.12E-11*R*R;

...
KA := A0 + A1*S + A2*S*S;
END; (procedure Surface)

PROCEDURE Static_Test;

VAR
I : Integer;
X1,X2,Static_D1,Static_D2 : Real;
BEGIN

X1 := ( 32*F_Safe*SQR(SQR(M_Mom) + SQR(A_Torq) ) )/(PI*S_Ys);
Static_D1 := Raise( X1, 0.33333);
X2 := ( 32*F_Safe*SQR(SQR(M_Torq) + SQR(A_Mom) ) )/(PI*S_Ys);
Static_D2 := Raise( X2, 0.33333);  

IF Tes_D1 < Static_D1 THEN
BEGIN
  Flag := 0;
  Writeln;
  Writeln('STATIC REQUIREMENTS NOT SATISFIED, THE SECTION');
  Writeln('OF MAX. MOMENT/I WITH ASSOCIATED TORQUE HAS DIA. = ',Tes_D1:6:4);
  Writeln('A DIAMETER = ',Static_D1:6:4,' IS NEEDED.');
END; {if}

IF No_Load > 1 THEN
BEGIN
  IF Tes_D2 < Static_D2 THEN
  BEGIN
    Flag := 0;
    Writeln;
    Writeln('STATIC REQUIREMENTS NOT SATISFIED, THE SECTION');
    Writeln('OF MAX. TORQUE/I WITH ASSOCIATED MOMENT HAS DIA. = ',Tes_D2:6:4);
    Writeln('A DIAMETER = ',Static_D2:6:4,' IS NEEDED.');
  END; {if}
  END; {if}

IF Flag = 1 THEN
BEGIN
  Writeln;
  Writeln('STATIC REQUIREMENTS HAVE BEEN SATISFIED.');
END; {if}

END; {procedure static_test}

PROCEDURE Final;
VAR
  ch : Char;
  I,IX,IY,Flag : Integer;
  Delx,Max_Dia,Temp : Real;
  Max_X,Max_Y,X,X1,X2,Y,Y1,Y2 : Real;
BEGIN
  IF (LShaft - Dist_Bear[No_Bear]) < (WB/2 + 0.125) THEN
    LShaft := Dist_Bear[No_Bear] + (WB/2 + 0.125);
  IF NumH <> 0 THEN
    BEGIN
      IF (LShaft - Dist_Bear[No_Bear]) < (WB/2 + 0.125) THEN
        Sec_Len[No_SecChDia] := Sec_Len[No_SecChDia] + (WB/2 + 0.125);
      END;
    IF Dist_Bear[1] < (WB/2 + 0.125) THEN
      BEGIN
        Delx := ((WB/2 + 0.125) - Dist_Bear[1]);
        LShaft := LShaft + Delx;
        FOR I := 1 TO No_Bear DO
          Dist_Bear[I] := Dist_Bear[I] + Delx;
        FOR I := 1 TO No_Load DO
          Dist_Load[I] := Dist_Load[I] + Delx;
        IF No_Key <> 0 THEN
          BEGIN
            FOR I := 1 TO No_Key DO
              Key_Loc[I] := Key_Loc[I] + Delx;
          END;
        IF No_Grv <> 0 THEN
          BEGIN
            FOR I := 1 TO No_Grv DO
              Grv_Loc[I] := Grv_Loc[I] + Delx;
          END;
        IF No_Hole <> 0 THEN
          BEGIN
            FOR I := 1 TO No_Hole DO
              Hole_Loc[I] := Hole_Loc[I] + Delx;
          END;
      END;
    END;
  Temp := -1;
  FOR I := 1 TO 10 DO
    BEGIN
      IF Dia[I] > Temp THEN
        BEGIN
          Temp := Dia[I];
          Max_Dia := Dia[I];
        END;
    END;
END;
Writeln;'  
PRELIMINARY DESIGN DISPLAY';
Writeln;
Writeln('DO YOU WANT A HARDCOPY OF THE DISPLAY Y/N ?');
Write('ANSWER >> '); 
Read( ch ); ch:= UpCase( ch ); 
IF ch = 'Y' THEN Flag:= 1 ELSE Flag:= 0;
EnterGraphic;
Max_X:= LShaft + 1;
Max_Y:= Max_Dia*5;
DefineWorld( 3, -1, 0, Max_X, Max_Y);
SelectWorld(3);
SelectWindow(1);
DrawSquare( -1, 0, Max_X, Max_Y, False);
SetLineStyle(1);
DrawLine( -1, Max_Y/2, Max_X, Max_Y/2);
Y1 := Max_Y/2 - (Max_Y/3);
DrawLine( 0, Y1, 0, Max_Y/2);
IX:= Round( 80/(Max_X + 1) );
GotoXY(IX,22); Write('0');
DrawLine( LShaft, Y1, LShaft, Max_Y/2);
IX:= Trunc( 80*(LShaft+1)/(Max_X+1) );
GotoXY(IX,22); Write(LShaft:5:3);
SetLineStyle(0);
IF NumH <> 0 THEN BEGIN 
X1:= 0.0;
X2:= 0.0;
FOR I:= 1 TO No_SecChDia DO BEGIN 
IF I > 1 THEN X1:= X1 + Sec_Len[I-1];
Y1:= Max_Y/2 + Dia[I]/2;
X2:= X2 + Sec_Len[I];
Y2:= Max_Y/2 - Dia[I]/2;
DrawSquare( X1, Y1, X2, Y2, False);
END; {for} 
END; {if} 
FOR I:= 1 TO No_Bear DO BEGIN 
X1:= Dist_Bear[I] - WB/2;
Y1:= Max_Y/2 + OD/2;
X2:= Dist_Bear[I] + WB/2;
Y2:= Max_Y/2 + BORE/2;
DrawSquare( X1, Y1, X2, Y2, True);
Y1:= Max_Y/2 - BORE/2;
Y2:= Max_Y/2 - OD/2;
DrawSquare( X1, Y1, X2, Y2, True);
END; {for}

FOR I:= 1 TO No_Load DO
BEGIN
  X1:= Dist_Load[I] - Width_Hub[I]/2;
  Y1:= Max_Y/2 + 1.3*Max_Dia;
  X2:= Dist_Load[I] + Width_Hub[I]/2;
  Y2:= Max_Y/2 - 1.3*Max_Dia;
  DrawSquare( X1, Y1, X2, Y2, False);
END; {for}

SetLineStyle(1);
Y:= Max_Y/2 + Max_Y/3;
FOR I:= 1 TO No_Bear DO
BEGIN
  DrawLine( Dist_Bear[I], Max_Y/2 + BORE/2, Dist_Bear[I], Y);
  IX:= Round( 80*(Dist_Bear[I]+1)/(Max_X + 1) ) - 2;
  IY:= 25 - Round( 25*Y/Max_Y );
  GotoXY( IX,IY ); Write( Dist_Bear[I]:5:3);
END; {for}

Y:= Max_Y/2 - Max_Y/3;
FOR I:= 1 TO No_Load DO
BEGIN
  DrawLine( Dist_Load[I], Max_Y/2, Dist_Load[I], Y);
  IX:= Round( 80*(Dist_Load[I]+1)/(Max_X + 1) ) - 2;
  IY:= 25 - Round( 25*Y/Max_Y );
  GotoXY( IX,IY ); Write( Dist_Load[I]:5:3);
END; {for}

IF No_Hole > 0 THEN
BEGIN
  FOR I:= 1 TO No_Hole DO
  BEGIN
    IX:= Round( 80*(Hole_Loc[I] + 1.1)/(Max_X + 1) );
    GotoXY( IX,13 ); Write('H');
  END; {for}
END; {if}

IF No_Grv > 0 THEN
BEGIN
  FOR I:= 1 TO No_Grv DO
  BEGIN
    IX:= Round( 80*(Grv_Loc[I] + 1.1)/(Max_X + 1) );
    GotoXY( IX,13 ); Write('G');
  END; {for}
END; {if}

IF No_Key > 0 THEN
BEGIN


FOR I := 1 TO No_Key DO
BEGIN
   IX:= Round( 80*(Key_Loc[I] + 1.1)/(Max_X + 1) );
   GotoXY( IX,13 ); Write('K');
END; {for}
END; {if}

GotoXY(2,24); Writeln('* NOTE: BEARINGS ARE ','VND,B1:2,Prt_Num);

IF Flag = 1 THEN
BEGIN
   SelectScreen(1);
   HardCopy( false,1 );
END; {if}

Repeat
Read( Kbd,ch );
Until ch = ', ';
LeaveGraphic;
END; {procedure Final}

BEGIN
IF Change_Prop = 1 THEN GOTO 2;
ClrScr;
Writeln( '**** STRESS CONCENTRATION **** ');
Writeln;
Repeat
Writeln('DO YOU WANT TO CHANGE THE NUMBER OR LOCATION OF GROOVES ? ');
Write('ANSWER Y OR N >> ');
Readln( ans ); ans:= UpCase( ans );
IF ans = 'Y' THEN
BEGIN
Writeln;
Write('ENTER THE NUMBER OF GROOVES >> ');
Readln( No_Grv );
Writeln;
IF No_Grv > 0 THEN
BEGIN
   FOR I := 1 TO No_Grv DO
   BEGIN
      Write('ENTER LOCATION (in.) GROOVE (',I,') >> ');
      Readln( Grv_Loc[I] );
   END; {for}
END; {if}
IF No_Grv > 0 THEN
BEGIN
   Writeln;
   Sum:= 0.0;
J := 0;
FOR I := 1 TO No_Grv DO
BEGIN
  Write('ENTER RADIUS OF GROOVE(',I,') > > ');
  Readln( Grv_Rad[I] );
  WHILE Sum < Grv_Loc[I] DO
  BEGIN
    J := J + 1;
    Sum := Sum + Sec_Len[J];
  END; {while}
  Writeln('THE DIAMETER AT THIS SECTION = 
',Dia[J]:5:4);
END; {for}
END; {if}
IF ( ans <> 'Y' ) AND ( ans <> 'N' ) THEN
Writeln('INCORRECT RESPONSE -- TRY AGAIN');
Until ( ans = 'Y' ) OR ( ans = 'N' );

Writeln;
Repeat
  Writeln('DO YOU WANT TO CHANGE THE NUMBER OR LOCATION OF KEYWAYS ?');
  Write('ANSWER Y OR N > > ');
  Readln( ans ); ans := UpCase( ans );
  IF ans = 'Y' THEN
  BEGIN
    Writeln; Write('ENTER THE NUMBER OF KEYWAYS > > ');
    Readln( No_Key );
    Writeln;
    FOR I := 1 TO No_Key DO
    BEGIN
      Write('ENTER LOCATION (in.) OF KEYWAY(',I,') > > ');
      Readln( Key_Loc[I] );
    END; {for}
  END; {if}
  IF No_Key > 0 THEN
  BEGIN
    Repeat
      Writeln('IS THE BRINELL HARDNESS OF THE MATERIAL 1) < 200 2) > 200 ?');
      Write('CHOICE > > ');
      Readln( Opt );
    CASE Opt OF
    '1': Keytb := 1.6;
    '2': Keytb := 2.0;
ELSE Writeln('ERROR: SELECTION INVALID -- TRY AGAIN');
END; {case}
Until ( Opt = '1' ) OR ( Opt = '2' );
END; {if}
IF ( ans <> 'Y' ) AND ( ans <> 'N' ) THEN
Writeln('INCORRECT RESPONSE -- TRY AGAIN');
Until ( ans = 'Y' ) OR ( ans = 'N' );

Writeln;
Repeat
Writeln('DO YOU WANT TO CHANGE THE NUMBER OR LOCATION OF HOLES ?');
Write('ANSWER Y OR N >> ');
Readln( ans ); ans:= UpCase( ans );
IF ans = 'Y' THEN
BEGIN
Writeln;
Write('ENTER THE NUMBER OF HOLES >> ');
Readln( No_Hole );
IF No_Hole > 0 THEN
BEGIN
FOR I:= 1 TO No_Hole DO
BEGIN
Write('ENTER LOCATION(in.) HOLE(' , I , ') >> ');
Readln( Hole_Loc[I] );
END; {for}
END; {if}
END; {if}
IF No_Hole > 0 THEN
BEGIN
Writeln;
FOR I:= 1 TO No_Hole DO
BEGIN
Write('ENTER THE DIAMETER OF HOLE(' , I , ') »');
Readln( Hole_D[I] );
END; {for}
END; {if}
IF ( ans <> 'Y' ) AND ( ans <> 'N' ) THEN
Writeln('INCORRECT RESPONSE -- TRY AGAIN.');
Until ( ans = 'Y' ) OR ( ans = 'N' );

Temp:= 1.0;
Temp1:= 1.0;
Temp2:= 1.0;

IF NumH <> 0 THEN Mod_FacKE;

IF No_Key <> 0 THEN Mod_FacKE1;
BEGIN
    M_Torq := 0.0;
    A_Mom := 0.0;
    Max_S2 := 0.0;
    Tes_D2 := 0.0;
END; (if)

2: ClrScr;
Writeln;
Writeln('ENDURANCE-LIMIT MODIFYING FACTORS KA, KB, KC, KE');
Writeln;
Writeln('THE SURFACE FACTOR (KA)');
Writeln('IS BASED ON THE SURFACE ROUGHNESS MEASURED IN MICROINCHES');
Write('ENTER THE SURFACE ROUGHNESS >> '); Readln( R );
Surface;
Write('KA >> ',KA:5:4);
Write;
Writeln('THE SIZE FACTOR KB >> ',KB:5:4);
Write;
Writeln('THIS BASED ON THE DIAMETER AT THE MAX. MOMENT');
Write;
Writeln('ENTER THE NUMBER OF DESIRED RELIABILITY, KC');
Write(' 1) 0.5  2) 0.90  3) 0.95  4) 0.99  5) 0.999 '); Write;
Readln( Choice );
CASE Choice OF
    '1': KC := 1.0;
    '2': KC := 0.897;
    '3': KC := 0.868;
    '4': KC := 0.814;
    '5': KC := 0.753;
ELSE
    BEGIN

Mod_FacKB;

IF (Tes_D1 = DS) OR (Tes_D1 = BORE) THEN
Mod_FacKE2;

Kf := 1;
IF Temp > Kf THEN Kf := Temp;
IF Temp1 > Kf THEN Kf := Temp1;
IF Temp2 > Kf THEN Kf := Temp2;
KE := 1/Kf;

IF No_Load > 1 THEN Mx_Torq;

IF No_Load = 1 THEN
BEGIN
    M_Torq := 0.0;
    A_Mom := 0.0;
    Max_S2 := 0.0;
    Tes_D2 := 0.0;
END; (if)
KC := 0.753;
WriteLn('INVALID SELECTION : KC = ',KC);
END
END; {case}
WriteLn;
WriteLn('BASED ON EFFECTS OF STRESS CONCENTRATION');
Write('THE ENDURANCE-LIMIT STRENGTH REDUCTION FACTOR KE ');
WriteLn(',KE:5:4);
WriteLn; WriteLn;
Write('ENTER THE DESIRED FACTOR OF SAFETY N >> ');
ReadLn(F_Safe);
IF (F_Safe < 1.0) OR (F_Safe > 3.0) THEN F_Safe := 1.5;
Flag := 1;
Static_Test;

IF Mod_Elast = 1.0E+07 THEN S_E := 0.7*KA*KB*KC*KE*S_Ult;
IF Mod_Elast = 3.0E+07 THEN S_E := KA*KB*KC*KE*S_Ult;

IF Max_S1 > 0.25*S_Ys THEN
BEGIN
  Y := (32*F_Safe*SQR(SQR(A_Torq/S_Ys) + SQR(M_Mom/S_E)))/PI);
  Cal_D1 := Raise(Y,0.33333);
END ELSE
BEGIN
  Y := (32*F_Safe*M_Mom/(PI*S_E));
  Cal_D1 := Raise(Y,0.33333);
END;

IF Tes_D1 < Cal_D1 THEN
BEGIN
  Flag := 0;
  WriteLn;
  WriteLn('FATIGUE REQUIREMENTS NOT SATISFIED, THE SECTION');
  WriteLn('OF MAX. MOMENT/I WITH ASSOCIATED TORQUE HAS A DIA. = ',Tes_D1:6:4);
  WriteLn('A DIAMETER OF ABOUT DIA. = ',Cal_D1:6:4,' IS NEEDED. ');
END;

IF No_Load > 1 THEN
BEGIN
  IF Max_S2 > 0.25*S_Ys THEN
  BEGIN
    Y := (32*F_Safe*SQR(SQR(M_Torq/S_Ys) + SQR(A_Mom/S_E)))/PI);
    Cal_D2 := Raise(Y,0.33333);
    IF Tes_D2 < Cal_D2 THEN
BEGIN
  Flag:= 0;
  Writeln;  
  Writeln('FATIGUE REQUIREMENTS NOT SATISFIED, ');
  Writeln('OF MAX. TORQUE/I WITH ASSOCIATED 
MOMENT HAS A DIA. = ',Tes_D2:6:4);
  Writeln('A DIAMETER OF ABOUT DIA. = ');
  Writeln(',Cal_D2:6:4,' IS NEEDED.');
  END; {if}
  END; {if}
END; {if}

IF Flag > 0 THEN
BEGIN
  Writeln;
  Writeln('FATIGUE REQUIREMENTS HAVE BEEN SATISFIED BY A ');
  Writeln('DIA. = ',Tes_D1:6:4);
  Final;
  Writeln('SECTION NO.',Bl:5,'DIAMETER(in)',B1:6,'SECTION 
LENGTH(in)');
  Writeln;
  FOR I:= 1 TO No_SecChDia DO
    Writeln(Bl:5,r,Bl:11,Dia[I]:6:4,Bl:15,Sec_Len[I]:6:4);
  END;

IF Flag = 0 THEN
BEGIN
  Writeln; Writeln;
  Assign(ChainFile, 'SubMain.chn');Chain(Chainfile);
  END; {if}
END. { end subprogm }
PROGRAM SubMain(input,output);
{$I typedef.sys}
{$I graphix.sys}
{$I kernel.sys}

LABEL
  1,2;

CONST
  PI = 3.141592;
  B1 = ' ';
  MX = 51;  {Number of equations or pts. along shaft}
  MAX = 52;  {No. of Eqn +1 Load Vector }
  Ftype = '.DAT';
  Max_Char = 6;

TYPE
  Real2 = ARRAY[1..2] OF Real;
  Real11= ARRAY[1..11] OF Real;
  Vals = ARRAY[0..MX] OF Real;
  Copy = ARRAY[0..MX] OF Real;
  RValues = ARRAY[1..10] OF Real;
  IVValues = ARRAY[1..10] OF Integer;
  Strings = String[Max_Char];
  Matrix = ARRAY[1..MX,1..MAX] OF Real;

VAR
  ans,Opt,Choice : Char;
  Catalog : Text;
  Filename: String[12];
  VND,Prt_Num : Strings;
  Change_Prop : Integer;
  S_E,F_Safe,Cal_D1,Cal_D2,Tes_D1,Tes_D2 : Real;
  Ktbgr,Ktbh,Ktbst,KA,KB,KE,Kf,M_Mom,M_Torq,A_Torq,A_Mom,-
  Max_S1,Max_S2 : Real;
  No_Grv,No_Key,No_Hole,NumH,Point,WhereToStart,Flag : Integer;
  BORE,OD,WB,FR,DS,DH,CR,Fillet,APower,D_Ratio,R_Ratio,-
  Keytb,RAD,q : Real;
  X,Y,R,Temp,Temp1,Temp2,Max_Y,Min_Y,CRD,V,BRel,NSpeed,Step1-
  ,Step2 : Real;
  Mod_Elast,S_Ys,S_Ult,LShaft,DX,Delta_Conc,TFx,DHours,-
  Sum_Torq,Sum : Real;
  Start,Finish,No_SecChDia,No_Load,Set_Up,Sect_Th,str_Ra-
  nise , -
  Compr_End : Integer;
  Last,II,JJ,Tick,Mark,MatSize,I,J,K,L,Cnt,No_Bear,No_Seg,MS : Integer;
  Bear_Pos : IVValues;
FUNCTION Sign(Number : Real):Real;
BEGIN
IF Number = 0.0
THEN Sign := 1
ELSE Sign := Abs(Number)/Number
END; {function sign}

FUNCTION Raise(Number,Power : Real):Real;
BEGIN
IF Number = 0.0
THEN IF Power = 0.0
      THEN Raise := 1.0
      ELSE Raise := 0.0
ELSE Raise := Sign(Number)*Exp(Power*Ln(Abs(Number)))
END; {function raise}

PROCEDURE Design_Area;

VAR
   ch : Char;
   X_OffSet,X : Real;
PROCEDURE Arrows;
VAR
XAxis_Len, HArrowLen, VArrowLen, HTail, VTail : Real;
Col_X, Col_X1, Row_Y, Row_Y1 : Integer;
BEGIN
SetLineStyle(0);
XAxis_Len := X_OffSet + (0.7*X_OffSet);
HArrowLen := XAxis_Len - (0.2*XAxis_Len);
HTail := 0.02*(Mid_Scr + 5);
VArrowLen := Mid_Scr + 5;
VTail := 0.07*XAxis_Len;
DrawLine( X_OffSet, Mid_Scr, XAxis_Len, Mid_Scr );
DrawLine( X_OffSet, Mid_Scr, X_OffSet, Mid_Scr+7 );
DrawLine( X_OffSet, Mid_Scr+7, X_OffSet+VTail, VArrowLen );
DrawLine( X_OffSet, Mid_Scr+7, X_OffSet-VTail, VArrowLen );
DrawLine( XAxis_Len, Mid_Scr, HArrowLen, Mid_Scr+HTail );

Col_X1 := Trunc( 80*X_OffSet/Max_LenX );
Row_Y1 := 25-Trunc( 25*(Mid_Scr+1)/Max_LenY );
GotoXY(Col_X1, Row_Y1); Write( 'Y' );
Col_X := Trunc( 80*XAxis_Len/Max_LenX );
Row_Y := 25-Trunc( 25*(Mid_Scr-4)/Max_LenY );
GotoXY(Col_X, Row_Y); Write( 'X' );
END; {procedure arrows}

BEGIN
EnterGraphic;
ResetWorlds;
ResetWindows;
Max_LenX := Trunc( LShaft+1 );
Max_LenY := 80;
DefineWorld(1,0,0,Max_LenX,Max_LenY);
SelectWorld(1);
SelectWindow(1);
S1 := 0;
S2 := Max_LenY - Trunc(0.8*Max_LenY);
DrawSquare( S1, S2, Max_LenX, Max_LenY, false );
Y_Txt := 25-Trunc( S2*25/Max_LenY );
GotoXY(30, Y_Txt); Writeln('* *** SHAFT DESIGN ***');
SetLineStyle(3);
S3 := Trunc( (Max.LenY - S2)/2 );
Mid_Scr := Max.LenY - S3;
X_OffSet := 0.1*Max.LenX;
DrawLine( S1, Mid_Scr, Max.LenX, Mid_Scr );
DrawLine( X_OffSet, Mid_Scr-20, X_OffSet, Mid_Scr+5 );
DrawLine( LShaft, Mid_Scr-5, LShaft, Mid_Scr+25 );

SetLineStyle(0);
FOR Tick:= 1 TO Max.LenX DO
BEGIN
   Mark := Tick;
   DrawLine( Mark, Mid_Scr+1, Mark, Mid_Scr-1 )
END; {for}
X_ColAxis := Trunc( 80*X_OffSet/Max.LenX );
Y_RowAxis := Trunc( 25*(Mid_Scr+10)/Max.LenY );
GotoXY( X_ColAxis, Y_RowAxis ); Write('0');
X_ColShaftEnd := Trunc( 80*LShaft/Max.LenX );
GotoXY( X_ColShaftEnd, 2 ); Write( LShaft:5:3 );

SetLineStyle(1);
Mark_Bear := Mid_Scr - 8;
Line_Len := S2 + Trunc( 0.5*S2 );
FOR I:= 1 TO No_Bear DO
BEGIN
   X := Dist_Bear[I];
   DrawLine( X, Mark_Bear, X, Mark_Bear+Line_Len );
   Col_X := Trunc( 80*X/Max.LenX );
   IF Col_X = 0 THEN Col_X:= 1;
   GotoXY( Col_X, 4 ); Write( X:5:3 );
END; {for}
GotoXY( 1, 2 ); Write('0');
Arrows;
Repeat
   Read( Kbd, ch )
Until ch = ' '; LeaveGraphic;
END; {procedure design_area}

PROCEDURE In_Matrix;

VAR
   I,J,K : Integer;

BEGIN
   FOR I:= 1 TO MatSize DO
      FOR J:= 1 TO MatSize+1 DO
         Stiff^I,J := 0; {load initial zeros into matrix}
   FOR K:= 3 TO MatSize-2 DO {load stiffness matrix as if ends free}
      BEGIN

Stiff^[K,K-2] := 1;
Stiff^[K,K-1] := -4;
Stiff^[K,K] := 6;
Stiff^[K,K+1] := -4;
Stiff^[K,K+2] := 1;
END; {for}

Stiff^[1,1] := 2;
Stiff^[1,2] := -4;
Stiff^[1,3] := 2;
Stiff^[2,1] := -2;
Stiff^[2,2] := 5;
Stiff^[2,3] := -4;
Stiff^[2,4] := 1;
Stiff^[MatSize-1,MatSize-3] := 1;
Stiff^[MatSize-1,MatSize-2] := -4;
Stiff^[MatSize-1,MatSize-1] := 5;
Stiff^[MatSize-1,MatSize] := -2;
Stiff^[MatSize,MatSize-2] := 2;
Stiff^[MatSize,MatSize-1] := -4;
Stiff^[MatSize,MatSize] := 2;

END;{procedure In_Matrix}

PROCEDURE Over_Write;

VAR
   I,J,Z,Shift : Integer;

BEGIN
   IF No_Bear > 1 THEN
      BEGIN
         IF ( Bear_Pos[1] = 0 ) AND ( Bear_Pos[No_Bear] < Trunc((LShaft-DX)/DX) ) THEN
            BEGIN
               Set_Up := 0;
               IF No_Bear > 1 THEN
                  BEGIN
                     SHIFT := 2;
                     FOR I:= 2 TO No_Bear DO
                        BEGIN
                           Z := Bear_Pos[I] - Shift;
                           Shift := Shift + 1;
                           Stiff^[Z,Z+2] := 0;
                           Stiff^[Z+1,Z+2] := 1; {CASE 1}
                           Stiff^[Z+1,Z+3] := 0;
                           Stiff^[Z+2,Z] := 0;
                           Stiff^[Z+2,Z+1] := 1;
                           Stiff^[Z+3,Z+1] := 0;
                        END; {for}
                  END; {i f}
            END;
         END; {i f}
      END; {i f}
   END;}
IF ( Bear_Pos[1] = 1 ) AND ( Bear_Pos[No_Bear] >= Trunc((LShaft-DX)/DX) ) THEN
BEGIN
  Set_Up := 1;
  IF No_Bear > 2 THEN
  BEGIN
    Shift := 3;
    FOR I:= 2 TO No_Bear-1 DO
    BEGIN
      Z := Bear_Pos[I] - Shift;
      Shift := Shift + 1;
      Stiff[Z,Z+2] := 0;
      Stiff[Z+1,Z+2] := 1;
      Stiff[Z+1,Z+3] := 0;
    END;
  END;
END; (if)

END; (if)

IF ( Bear_Pos[1] = 0 ) AND ( Bear_Pos[No_Bear] >= Trunc((LShaft-DX)/DX) ) THEN
BEGIN
  Set_Up := 0;
  IF No_Bear > 2 THEN
  BEGIN
    Shift := 2;
    FOR I:= 2 TO No_Bear-1 DO
    BEGIN
      Z := Bear_Pos[I] - Shift;
      Shift := Shift + 1;
      Stiff[Z,Z+2] := 0;
      Stiff[Z+1,Z+2] := 1;
      Stiff[Z+1,Z+3] := 0;
    END;
  END;
END; (if)

END; (if)

IF ( Bear_Pos[1] > 1 ) AND ( Bear_Pos[No_Bear] < Trunc((LShaft-DX)/DX) ) THEN
BEGIN
  Set_Up := 2;
  Shift := 1;
  FOR I:= 1 TO No_Bear DO
BEGIN
  Z := Bear_Pos[I] - Shift;
  Shift := Shift + 1;
  Stiff[Z,Z+2] := 0;
  Stiff[Z+1,Z+2] := 1; {CASE 4}
  Stiff[Z+1,Z+3] := 0;
  Stiff[Z+2,Z] := 0;
  Stiff[Z+2,Z+1] := 1;
  Stiff[Z+3,Z+1] := 0;
END; {for}
END; {if}

IF ( Bear_Pos[1] = 1 ) AND ( Bear_Pos[No_Bear] < Trunc((LShaft-DX)/DX) ) THEN
BEGIN
  Set_Up := 1;
  Shift := 3;
  FOR I:= 2 TO No_Bear DO
    BEGIN
      Z := Bear_Pos[I] - Shift;
      Shift := Shift + 1;
      Stiff[Z,Z+2] := 0;
      Stiff[Z+1,Z+2] := 1; {CASE 5}
      Stiff[Z+1,Z+3] := 0;
      Stiff[Z+2,Z] := 0;
      Stiff[Z+2,Z+1] := 1;
      Stiff[Z+3,Z+1] := 0;
    END; {for}
END; {if}

IF ( Bear_Pos[1] > 1 ) AND ( Bear_Pos[No_Bear] >= Trunc((LShaft-DX)/DX) ) THEN
BEGIN
  Set_Up := 2;
  Shift := 1;
  FOR I:= 1 TO No_Bear-1 DO
    BEGIN
      Z := Bear_Pos[I] - Shift;
      Shift := Shift + 1;
      Stiff[Z,Z+2] := 0;
      Stiff[Z+1,Z+2] := 1; {CASE 6}
      Stiff[Z+1,Z+3] := 0;
      Stiff[Z+2,Z] := 0;
      Stiff[Z+2,Z+1] := 1;
      Stiff[Z+3,Z+1] := 0;
    END; {for}
END; {if}
END; {End The Main IF Statement}
IF ( Bear_Pos[1] = 0 ) OR ( Bear_Pos[1] = 1 ) THEN
BEGIN
    Stiff^[1,1] := 5;
    Stiff^[1,3] := 1;
    Stiff^[2,1] := -4;
    Stiff^[2,2] := 6;
END; {if}

IF Bear_Pos[No_Bear] >= Trunc( (LShaft-DX)/DX ) THEN
BEGIN
    Stiff^[MatSize,MatSize] := 5;
    Stiff^[MatSize,MatSize-2] := 1;
    Stiff^[MatSize-1,MatSize] := -4;
    Stiff^[MatSize-1,MatSize-1] := 6;
END; {if}
END; {procedure Over_Write}

PROCEDURE Inertia;

LABEL
1,2;

VAR
    I,J,K,L,Off_Set : Integer;

BEGIN

    FOR I:= 1 TO No_Load DO
        Load[I] := (Load[I]*Delta_Cons)/(MomI[1]*Mod_Elast);

    CASE Set_Up OF
        0: Off_Set := 1; {depending on the starting end condition}
        1: Off_Set := 2; {the load location in the matrix must}
        ELSE Off_Set := 0; {have some initial offset, even of 0 }
    END; {case}

    J := 1;

    IF ( Set_Up = 0 ) OR ( Set_Up = 1 ) THEN
BEGIN
    FOR I:= 1 TO No_Load DO {find the correct location in the matrix}
        BEGIN {for the load}
            IF J < No_Bear THEN
            BEGIN
                1: IF Dist_Bear[J+1] < Dist_Load[I] THEN
                    BEGIN
                        Off_Set := Off_Set + 1;
                        J := J + 1;
                    END;
            END;
        END;
    END;
END; {procedure Inertia}
END; {if}
IF ( J < No_Bear ) AND ( Dist_Bear[J+1] < Dist_Load[I] ) THEN GOTO 1;
    Load_Loc[I] := Load_Loc[I] - Off_Set;
END; {if}
END; {for}
END; {if}

IF ( Set_Up <> 0 ) AND ( Set_Up <> 1 ) THEN BEGIN
    FOR I:= 1 TO No_Load Do
        BEGIN
            {find the correct location in the matrix for
            the load}
            IF J <= No_Bear THEN
                BEGIN
                    IF Dist_Bear[J] < Dist_Load[I] THEN
                        BEGIN
                            Off_Set := Off_Set + 1;
                            J := J + 1;
                        END; {if}
                    IF ( J <= No_Bear ) AND ( Dist_Bear[J] < Dist_Load[I] ) THEN GOTO 2;
                    Load_Loc[I] := Load_Loc[I] - Off_Set;
                END; {if}
            END; {for}
        END; {if}
    FOR I:= 1 TO No_Load DO
        BEGIN
            L := Load_Loc[I];
            Stiff[L,MatSize+1] := Load[I];
        END; {for}
END; { procedure inertia }

PROCEDURE Invert;
LABEL 1;
VAR
    R,Temp : Real;
    I,J,K : Integer;
BEGIN
    FOR K:= 1 TO MatSize DO
        BEGIN
            Temp:= Stiff[K,K];
            FOR J:= 1 TO MatSize+1 DO
                Stiff[K,J]:= Stiff[K,J]/Temp;
            FOR I:= 1 TO MatSize DO
                BEGIN

IF I=K THEN GOTO 1;
BEGIN
  R := Stiff^[I,K];
  FOR J := 1 TO MatSize+1 DO
    Stiff^[I,J] := Stiff^[I,J] - R*Stiff^[K,J];
END; (for)
END; (procedure invert)

PROCEDURE Copy_Deflect;

LABEL 1;
VAR
  I,J,K : Integer;
BEGIN
  FOR I := 0 TO MX-1 DO
    D[I] := 0;
  I := 1;
  J := -1;
  CASE Bear_Pos[1] OF
    0:
      BEGIN
        D[0] := 0;
        I := 2;
        J := 0;
      END;
    1:
      BEGIN
        D[0] := 0;
        D[1] := 0;
        J := 1;
        I := 2;
      END;
  END; (case of)

  MS := MatSize;
  IF Bear_Pos[No_Bear] = Trunc( LShaft/DX ) THEN MS := MatSize+1;
  IF Bear_Pos[No_Bear] = Trunc( (LShaft-DX)/DX ) THEN MS := MatSize+2;
  FOR K := 1 TO MS DO
    BEGIN
      J := J + 1;
      IF ( J = Bear_Pos[I] ) AND ( I <= No_Bear ) THEN
        BEGIN
          D[J] := 0;
        END;
      IF J = Bear_Pos[I] THEN D[J] := 0;
    END;
END; (procedure Copy_Deflect)
\[ J := J + 1; \]
\[ I := I + 1; \]
\[ \text{END; \{if\}} \]
\[ D[J] := \text{Stiff}^{\text{[K,MatSize+1]}}; \]
\[ \text{END; \{for\}} \]
\[ \text{END; \{procedure copy_deflect\}} \]

\text{PROCEDURE Beep;} \]
\text{VAR} \]
\text{ch : Char;} \]
\text{BEGIN} \]
\text{Sound(1000);} \]
\text{Delay(225);} \]
\text{NoSound;} \]
\text{Delay(125);} \]
\text{Sound(1000);} \]
\text{Delay(225);} \]
\text{NoSound;} \]
\text{Writeln; Writeln('CALCULATIONS FINISHED, PRESS SPACE BAR TO CONTINUE.');} \]
\text{Repeat} \]
\text{Read( Kbd,ch );} \]
\text{Until ch=' ');} \]
\text{END; \{procedure beep\}} \]

\text{PROCEDURE TOR;} \]
\text{VAR} \]
\text{I,J,L : Integer;} \]
\text{Sum_Torq : Real;} \]
\text{BEGIN} \]
\text{FOR I:= 0 TO MX-1 DO} \]
\text{Torque[I]:= 0;} \]
\text{1:= 1;} \]
\text{J:= 2;} \]
\text{Sum_Torq:= Torq[1];} \]
\text{FOR I:= 0 TO MX-1 DO} \]
\text{BEGIN} \]
\text{IF L < No_Load THEN} \]
\text{BEGIN} \]
\text{IF (I \geq \text{Round(Dist_Load[L]/DX)}) AND (I \leq \text{Round(Dist_Load[L+1]/DX)}) THEN Torque[I]:= Sum_Torq;} \]
\text{END; \{if\}} \]
\text{IF J \leq No_Load THEN} \]


BEGIN
  IF I = Round(Dist_Load[J]/DX) THEN
    BEGIN
      Sum_Torq := Sum_Torq + Torq[J];
      J := J + 1;
      L := L + 1;
    END; {if}
  END; {if}
END; {for}

PROCEDURE Vshear;

VAR
  I, LL, KK, II, JJ : Integer;
  Const1, TM2, TM1, T, TP1, TP2 : Real;

BEGIN
  forces and the reactions
  Const1 := 2*DX*DX*DX;
  J := 1;

  FOR I := 0 TO MX-1 DO
    Shear[I] := 0.0;

  FOR I := 2 TO MX-3 DO
    BEGIN
      TM2 := D[I-2]*(1);
      TM1 := D[I-1]*(2);
      T := D[I]*0;
      IF Round( Dist_Bear[I]/DX ) = 1 THEN BEGIN
        Shear[I] := -(TM2+TM1+T-TP1+TP2)*(
          (Mod_Elast*MomI[J])/Const1 );
      END; {for}
        (Mod_Elast*MomI[J])/Const1 );
      IF Round( Dist_Bear[I]/DX ) = 1 THEN
        BEGIN
          Shear[2] := Shear[3];
          Shear[1] := Shear[3];
        END;
    END;
  Shear[0] := Shear[1];
  Shear[MX-2] :=
\[-(D(MX-4)*(-1)+D(MX-3)*(2)+D(MX-2)*(-1)+D(MX-1)*(-2))*(\text{Mod Elast} \cdot \text{MomI}[J]) / \text{Const1} \]

\[
\begin{align*}
\text{IF Round}( \text{Dist_Bear}[\text{No_Bear}]/DX) &= \text{Round}( (\text{LShaft}-DX)/DX) \\
\text{THEN} \\
\text{BEGIN} \\
\text{Shear}[MX-3] &= \text{Shear}[MX-4]; \\
\text{Shear}[MX-2] &= \text{Shear}[MX-4]; \\
\text{END;}
\end{align*}
\]

\[
\begin{align*}
\text{Shear}[MX-1] &= \text{Shear}[MX-2]; \\
\text{LL} &= 1; \\
\text{KK} &= \text{No_Load};
\end{align*}
\]

\[
\begin{align*}
\text{IF Dist_Load}[1] &< \text{Dist_Bear}[1] \text{ THEN} \\
\text{BEGIN} \\
\text{J} &= \text{Round}( \text{Dist_Load}[1]/DX) ; \\
\text{Shear}[J] &= \text{Shear}[J+1]; \\
\text{LL} &= 2; \\
\text{IF J} &> 0 \text{ THEN} \\
\text{BEGIN} \\
\text{FOR K} &= 0 \text{ TO J-1 DO} \\
\text{Shear}[K] &= 0; \\
\text{END; } \{\text{if}\} \\
\text{END; } \{\text{if}\}
\end{align*}
\]

\[
\begin{align*}
\text{IF Dist_Load}[\text{No_Load}] &> \text{Dist_Bear}[\text{No_Bear}] \text{ THEN} \\
\text{BEGIN} \\
\text{J} &= \text{Round}( \text{Dist_Load}[\text{No_Load}]/DX) ; \\
\text{Shear}[J] &= -1 \cdot \text{Shear}[J-1]; \\
\text{KK} &= \text{No_Load} -1; \\
\text{IF J} &< MX-1 \text{ THEN} \\
\text{BEGIN} \\
\text{FOR K} &= J+1 \text{ TO MX-1 DO} \\
\text{Shear}[K] &= 0; \\
\text{END; } \{\text{if}\} \\
\text{END; } \{\text{if}\}
\end{align*}
\]

\[
\begin{align*}
\text{IF LL} &\leq KK \text{ THEN} \\
\text{BEGIN} \\
\text{FOR I} &= LL \text{ TO KK DO} \\
\text{BEGIN} \\
\text{J} &= \text{Round}( \text{Dist_Load}[I]/DX) ; \\
\text{Shear}[J] &= \text{Shear}[J+1] - \text{Shear}[J-1]; \\
\text{END; } \{\text{for}\} \\
\text{END; } \{\text{if}\}
\end{align*}
\]

\[
\begin{align*}
\text{II} &= 1; \\
\text{JJ} &= \text{No_Bear}; \\
\text{IF Dist_Bear}[1] &< \text{Dist_Load}[1] \text{ THEN} \\
\text{BEGIN} \\
\text{J} &= \text{Round}( \text{Dist_Bear}[1]/DX) ;
\end{align*}
\]
Shear[J] := Shear[J+1];
II := 2;
IF J > 0 THEN
BEGIN
FOR K := 0 TO J-1 DO
Shear[K] := 0;
END; (for)
END; (if)

IF Dist_Bear[No_Bear] > Dist_Load[No_Load] THEN
BEGIN
J := Round( Dist_Bear[No_Bear]/DX );
Shear[J] := -1*Shear[J-1];
JJ := No_Bear -1;
IF J < MX-1 THEN
BEGIN
FOR K := J+1 TO MX-1 DO
Shear[K] := 0;
END; (for)
END; (if)

IF II <= JJ THEN
BEGIN
FOR I := II TO JJ DO
BEGIN
J := Round( Dist_Bear[I]/DX );
Shear[J] := Shear[J+1] - Shear[J-1];
END; (if)
END; (for)
END; (procedure VShear)

PROCEDURE TMoment;

VAR
Const2, TM1, T, TP1 : Real;
I, J : Integer;

BEGIN
Const2 := DX*DX;
J := 1;

FOR I := 0 TO MX-1 DO
Moment[I] := 0.0;

FOR I := 1 TO MX-2 DO
BEGIN
TM1 := D[I-1];
T := D[I]*(-2);
TP1 := D[I+1];
Moment[I] := -(TM1 + T + TP1) * (Mod_Elast*MomI[J])/Const2);
END; (for)
Moment[0]:= 0;
IF Round( Dist_Bear[1]/DX ) = 1 THEN Moment[1]:= 0;
Moment[MX-1]:= 0;
IF Round( Dist_Bear[No_Bear]/DX ) = Round( (LShaft - DX)/DX ) THEN Moment[MX-2]:= 0;
END; {procedure Vmoment}

PROCEDURE HDCopy;

VAR
  Ch : Char;
  Prt : Text;
  I : Integer;
  W : Real;

BEGIN
  ClrScr;
  Writeln('HARDCOPY OF LOAD AND STRESS ON THE SHAFT');
  Writeln;
  Repeat
    Writeln('
    TURN ON THE PRINTER, AND PRESS THE SPACE
    BAR TO CONTINUE...');
    Read( Kbd,Ch )
    Until Ch = ' ';
    Assign( Prt, 'LST:' );
    Rewrite( Prt );
    Writeln( Prt,Bl:12,'*** LOAD AND STRESS DISTRIBUTION
    ON THE SHAFT ***');
    Writeln( Prt );
    Writeln( Prt );
    Writeln(Prt,'HUB
    Writeln( Prt );
    FOR I:= 1 TO No_Load DO
      Writeln(Prt,B1:3,I,B1:7,Dist_Load[I]:5:3,B1:7,CLoad[I]:7:2,B1:6,Torq[I]:7:2,B1:6,Thrust[I]:7:2);
      Writeln( Prt );
      Writeln(Prt,'DIST.(In)',B1:5,'DEFLECTION(In.)',B1:5,'SHEAR(lb)',B1:5,'MOMENT(lb-in)',B1:5,'TORQUE(lb-in)');
      Writeln( Prt );
      FOR I:= 0 TO MX-1 DO
        BEGIN
          W:= I*DX;
          Writeln(Prt,B1:2,W:5:3,B1:9,D[I]:8:6,B1:8,Shear[I]:8:2-,B1:8,Moment[I]:8:2,B1:10,Torq[I]:8:2);
        END;
    END;
    Close( Prt );
  END; {procedure hardcopy}

PROCEDURE Max_MinSH;
VAR
  I : Integer;
BEGIN
  Temp1 := 100000.0;
  Temp2 := -100000.0;
  FOR I := 0 TO MX-1 DO
    BEGIN
      IF Shear[I] >= Temp2 THEN
        BEGIN
          Max_Y := Shear[I];
          Temp2 := Max_Y;
        END;
      IF Shear[I] <= Temp1 THEN
        BEGIN
          Min_Y := Shear[I];
          Temp1 := Min_Y;
        END;
    END; {for}
END; {procedure Max_MinSH}

PROCEDURE Min_MaxMom;

VAR
  I : Integer;
BEGIN
  Temp1 := 100000.0;
  Temp2 := -100000.0;
  FOR I := 0 TO MX-1 DO
    BEGIN
      IF Moment[I] >= Temp2 THEN
        BEGIN
          Max_Y := Moment[I];
          Temp2 := Max_Y;
        END;
      IF Moment[I] <= Temp1 THEN
        BEGIN
          Min_Y := Moment[I];
          Temp1 := Min_Y;
        END;
    END; {for}
END; {procedure Min_MaxMom}

PROCEDURE SH_Dia;
LABEL
  1;

VAR
  ch : Char;
I,J,K,L,II,JJ,IX,IY,Flag : Integer;
X,Y,XX,Sfx,Sfy,Bottom,Top,Tick,Mark,Range,Temp1,Temp2 : Real;
BEGIN
EnterGraphic;
GotoXY(1,1);
Writeln('SHEAR DIAGRAM');
Writeln;
Writeln('DO YOU WANT A HARDCOPY OF THE DIAGRAM Y/N ?');
Write('ANSWER >> '); Read( ch ); ch:= UpCase( ch );
IF ch = 'Y' THEN Flag:= 1 ELSE Flag:= 0;
ClearScreen;
Bottom:= 0.125*(Min_Y - Max_Y); { assume min_y is zero or
negative no absolute value }
Top:= Abs( Bottom );
DefineWorld(2, -1, Min_Y+Bottom, LShaft+2, Max_Y+Top);
SelectWorld(2);
SelectWindow(1);
DrawSquare( -1, Min_Y+Bottom, LShaft+2, Max_Y+Top, false);
SetLineStyle(1);
DrawLine( 0, Min_Y, LShaft, Min_Y);
DrawLine( 0, Min_Y, 0, Max_Y);
DrawLine( 0, 0, LShaft, 0);
SetLineStyle(0);
Tick:= LShaft/10;
XX:= Abs( 0.25*Bottom );
FOR I:= 1 TO 10 DO
BEGIN
Mark:= Tick*I;
DrawLine( Mark, Min_Y, Mark, Min_Y+XX );
END; { for }
Range:= LShaft+2.3;
Sfx:= 80/Range;
FOR I:= 1 TO 10 DO
BEGIN
X:= 0.4+Tick*I;
IX:= Round(Sfx*X);
Mark:= Tick*I;
GotoXY(IX,24);
WriteXY(Mark:4:1);
END; { for }
Tick:= Round( (Max_Y-Min_Y)/10 );
FOR I:= 1 TO 10 DO
BEGIN
    Mark:= Min_Y + Tick*I;
    DrawLine(0, Mark, 0.02*LShaft, Mark);
END;

Range:= (Max_Y + Top) - (Min_Y + Bottom);
Sfy:= 25/Range;

FOR I:= 1 TO 10 DO
BEGIN
    Y:= Abs( Bottom ) + I*Tick;
    IY:= 25 - Trunc( Sfy*Y );
    Mark:= Min_Y + I*Tick;
    GotoXY( 1, IY ); Write( Mark:4:1 );
END; {for}

L:= 1;
K:= 1;
II:= No_Bear;
JJ:= No_Load;
Temp1:= 0;

BEGIN
    X:= Dist_Bear[1];
    J:= Round(Dist_Bear[1]/DX);
    DrawLine(X,0,X,Shear[J+1]);
    DrawLine(X,Shear[J+1],(J+1)*DX,Shear[J+1]);
    Temp1:= (J + 1)*DX;
    K:= K + 1;
    Temp2:= Shear[J+1];
END;

BEGIN
    X:= Dist_Load[1];
    J:= Round(Dist_Load[1]/DX);
    DrawLine(X,0,X,Shear[J+1]);
    DrawLine(X,Shear[J+1],(J+1)*DX,Shear[J+1]);
    Temp1:= (J + 1)*DX;
    L:= L + 1;
    Temp2:= Shear[J+1];
END;

IF Dist_Bear[No_Bear] > Dist_Load[No_Load] THEN
BEGIN
    X:= Dist_Bear[No_Bear];
    I:= Round(Dist_Bear[No_Bear]/DX);
    DrawLine( (I-1)*DX,Shear[I-1],X,Shear[I-1]);
    DrawLine(X,0,X,Shear[I-1]);
    Last:= I;
    II:= No_Bear-1;
END;

IF Dist_Load[No_Load] > Dist_Bear[No_Bear] THEN
BEGIN
X:= Dist_Load[No_Load];
I:= Round(Dist_Load[No_Load]/DX);
DrawLine( (I-1)*DX,Shear[I-1],X,Shear[I-1]);
DrawLine(X,0,X,Shear[I-1]);
Last:= I;
JJ:= No_Load-1;
END;

FOR I:= J+1 TO Last-1 DO
BEGIN
IF( I = Round(Dist_Bear[K]/DX) ) AND ( K <= JJ ) THEN
BEGIN
DrawLine( (I-1)*DX,Shear[I-1],I*DX,Shear[I-1] );
DrawLine( I*DX-0.001,Shear[I-1],I*DX+0.001,Shear[I+1] );
DrawLine( I*DX,Shear[I+1],(I+1)*DX,Shear[I+1] );
Temp1:= (I+1)*DX;
Temp2:= Shear[I+1];
K:= K + 1;
GOTO 1;
END;
IF( I = Round(Dist_Load[L]/DX) ) AND ( L <= JJ ) THEN
BEGIN
DrawLine( (I-1)*DX,Shear[I-1],I*DX,Shear[I-1] );
DrawLine( I*DX-0.001,Shear[I-1],I*DX+0.001,Shear[I+1] );
DrawLine( I*DX,Shear[I+1],(I+1)*DX,Shear[I+1] );
Temp1:= (I+1)*DX;
Temp2:= Shear[I+1];
L:= L + 1;
GOTO 1;
END;
DrawLine(Temp1,Temp2,I*DX,Shear[I]);
Temp1:= I*DX;
Temp2:= Shear[I];
END;{for}

GotoXY( 78,10); Write('S');
GotoXY( 78,11); Write('H');
GotoXY( 78,12); Write('E');
GotoXY( 78,13); Write('A');
GotoXY( 78,14); Write('R');

IF Flag = 1 THEN
BEGIN
SelectScreen(1);
HardCopy( false,1 );
END; {if}

Repeat
Read( Kbd, ch )
Until ch = ' ';

END; {procedure SH_Dia}

PROCEDURE Mom_Dia;

VAR
    ch : Char;
    I,J,IX,IY,Flag : Integer;
    X,Y,XX,Sfx,Sfy,Temp1,Temp2 : Real;
    Tick,Mark,Bottom,Top,Range : Real;

BEGIN
    ClearScreen;
    GotoXY(1,1);
    Writeln(' MOMEMT DIAGRAM');
    Writeln;
    Writeln('DO YOU WANT A HARDCOPY OF THE DIAGRAM Y/N ?');
    Write('ANSWER >> ');
    Read( ch ); ch:= UpCase( ch );
    IF ch = 'Y' THEN Flag:= 1 ELSE Flag:= 0;
    ClearScreen;
    Bottom:= Abs(0.125*(Max_Y-Min_Y));
    Top:= Abs(0.125*(Max_Y-Min_Y));
    DefineWorld(3,-1.1,Min_Y-Bottom,LShaft+1.2,Max_Y+Top);
    SelectWorld(3);
    SelectWindow(1);

    DrawSquare(-1.1,Min_Y-Bottom,LShaft+1.2,Max_Y+Top,false);

    SetLineStyle(1);
    DrawLine(0,Min_Y,LShaft,Min_Y);
    DrawLine(0,Min_Y,0,Max_Y);
    DrawLine(0,0,LShaft,0);

    Tick:= LShaft/10;
    Range:= LShaft+2.3;
    Sfx:= 80/Range;
    XX:= Abs(0.25*Bottom);
    SetLineStyle(0);

    FOR I:=1 TO 10 DO
        BEGIN
            Mark:= Tick*I;
            DrawLine( Mark, Min_Y, Mark, Min_Y+XX );
            X:= 1 + Tick*I;
            IX:= Round( Sfx*X );
            GotoXY( IX,24 ); Write( Mark:4:1 );
        END; {for}

    Tick:= Round( (Max_Y-Min_Y)/10 );
FOR I:= 1 TO 10 DO
    BEGIN
        Mark:= Min_Y + Tick*I;
        DrawLine( 0, Mark, 0.02*LShaft, Mark );
    END; {for

Range:= (Max_Y + Top) - (Min_Y - Bottom);
Sfy:= 25/Range;
    Bottom:= Bottom - 0.125*Bottom;

FOR I:= 1 TO 10 DO
    BEGIN
        Y:= Bottom + I*Tick;
        IY:= 25 - Round( Y*Sfy );
        Mark:= Min_Y + I*Tick;
        GotoXY( 1,IY ); Write( Mark:4:1 );
    END;

Temp1:= 0;
Temp2:= 0;
FOR I:= 0 TO MX-1 DO
    BEGIN
        DrawLine( Temp1, Temp2, I*DX, Moment[I] );
        Temp1:= I*DX;
        Temp2:= Moment[I];
    END; {for

GotoXY( 78,9); Write('M');
GotoXY( 78,10); Write('O');
GotoXY( 78,11); Write('M');
GotoXY( 78,12); Write('E');
GotoXY( 78,13); Write('N');
GotoXY( 78,14); Write('T');

IF Flag = 1 THEN
    BEGIN
        SelectScreen(1);
        HardCopy( false,1 );
    END; {if

Repeat
    Read( Kbd, ch )
Until ch = ' ';
LeaveGraphic;
END; {procedure Mom_Dia}

PROCEDURE Bear_Select;

LABEL
    1;
VAR
ans : Char;
I,J : Integer;
Forc_E,Forc_E1,Forc_E2,Forc_Max,Part_1,Part_2 : Real;
A,Power_1,Power_2,Exponent : Real;

BEGIN
Forc_E:= 0.0;
Forc_Max:= 0.0;
FOR I:= 1 TO No_Bear DO
BEGIN
J:= Round( Dist_Bear[I]/DX );
IF Bear_Type[I] = 0 THEN A:= 10/3;
IF Bear_Type[I] = 1 THEN A:= 3;
IF I = 1 THEN
BEGIN
Forc_E1:= V*Abs(Shear[J]);
Forc_E2:= 0.5*V*Abs(Shear[J]) +
1.4*Abs(Axial_Thr[1])
END; {if}
IF I = No_Bear THEN
BEGIN
Forc_E1:= V*Abs(Shear[J]);
Forc_E2:= 0.5*V*Abs(Shear[J]) +
1.4*Abs(Axial_Thr[2])
END; {if}
IF ( I <> 1 ) AND ( I <> No_Bear ) THEN
BEGIN
Forc_E1:= V*Abs(Shear[J]);
Forc_E2:= 0.5*V*Abs(Shear[J])
END; {if}
IF Forc_E1 > Forc_E2 THEN Forc_E:= Forc_E1 Else Forc_E:= Forc_E2;

IF Forc_E > Forc_Max THEN
BEGIN
Forc_Max:= Forc_E;
Exponent:= A;
END; {if}
END; {for}

Part_1:= ( DHours*NSpeed )/6840000.0;
Part_2:= Ln( 1/BRel );
Power_1:= 1/A;
Power_2:= 1/(1.17*A);

CRD:= (Forc_Max*Raise(Part_1,Power_1)
)/Raise(Part_2,Power_2);
Assign( Catalog,Filename );
Reset( Catalog );
WHILE NOT EOF( Catalog ) DO
BEGIN
Readln(Catalog,VND,Prt_Num,BORE,OD, WB,FR,DS,DH,CR);
    IF ( CR > CRD ) AND ( BORE >= Dia[1] ) THEN GOTO 1;
END; {while}
WriteLn;
WriteLn('WARNING: NO SUITABLE BEARING FOUND');
WriteLn('USING THE LARGEST BEARING IN THE CURRENT CATALOG');
1:Close( Catalog );
WriteLn;
WriteLn('BEARING SELECTED : VENDOR: ',VND,' PART NUM: ',Prt_Num);
WriteLn;
WriteLn('BEARING LOAD RATING >> ',CR:7:2);
WriteLn;
WriteLn('CALCULATED LOAD RATING >> ',CRD:7:2);
WriteLn;
WriteLn('PRESS SPACE BAR TO CONTINUE ...');
Repeat
Read( Kbd,ans )
Until ans = ' '
END; {procedure Bear_Select}

PROCEDURE Thick_Section;

VAR
    ch : Char;
    Top, Middle, WBHalf, Dist, DistA, DistB : Real;
    X, XA, XB, YA, Cnt, Fudge : Integer;

BEGIN
    Top:= BORE*10;
    Middle:= Top - 2.5*BORE;
    WBHalf:= WB/2.0;
    EnterGraphic;
    ResetWorlds;
    ResetWindows;
    DefineWorld(1, -1, 0, LShaft+1, BORE*10);
    SelectWorld(1);
    SelectWindow(1);
    DrawSquare(-1, 0, LShaft+1, Top, False);
    DrawSquare(0, Top-2*BORE, LShaft, Top-3*BORE, False);
    SetLineStyle(4);
    FOR I:= 1 TO No_Bear DO
        BEGIN
            DrawLine(Dist_Bear[I], Middle+BORE, Dist_Bear[I], Middle-BORE);
            DrawLine(Dist_Bear[I] - WBHalf, Middle, Dist_Bear[I] + WBHalf, Middle);
        END; {for}
DistA := Trunc( Dist_Bear[1]/2.0 ) + 1;
DistB := ( LShaft - Dist_Bear[No_Bear] )/2.0 +
Dist_Bear[No_Bear] + 1;

XA := Round( (80*DistA)/(LShaft+2) );
XB := Round( (80*DistB)/(LShaft+2) );
YA := 25 - Round( 25*(Middle - BORE)/(10*BORE) );

IF Dist_Bear[1] = 0 THEN Fudge := 0 ELSE Fudge := 1;
IF Dist_Bear[1] = 0 THEN Cnt := 1 ELSE Cnt := 2;

IF Dist_Bear[1] <> 0 THEN
BEGIN
GotoXY( XA, YA ); Write( '1' );
END;
IF LShaft - Dist_Bear[No_Bear] <> 0 THEN
BEGIN
GotoXY( XB, YA ); Write( No_Bear+Fudge );
END;

FOR I := 2 TO No_Bear DO
BEGIN
Dist := ( Dist_Bear[I] - Dist_Bear[I-1] )/2.0 +
Dist_Bear[I-1] + 1;
X := Round( (80*Dist)/(LShaft+2) );
GotoXY( X, YA ); Write( Cnt );
Cnt := Cnt + 1;
END; {for}

FOR I := 1 TO No_Bear DO
BEGIN
X := Round( 80*(Dist_Bear[I] + 1)/(LShaft + 2) );
YA := 25 - Round( 25*(Middle + BORE)/(10*BORE) );
GotoXY( X, YA ); Write( 'B' );
END; {for}

IF No_Grv <> 0 THEN
BEGIN
SetLineStyle(4);
FOR I := 1 TO No_Grv DO
BEGIN
DrawLine(Grv_Loc[I], Middle+BORE, Grv_Loc[I], Middle-BORE/2);
X := Round( 80*(Grv_Loc[I] + 1)/(LShaft+2) ) - 1;
YA := 25 - Round( 25*(Middle+BORE)/(10*BORE) );
GotoXY( X, YA ); Write('GRV');
END; {for}
END; {if}
IF No_Key <> 0 THEN
BEGIN
FOR I := 1 TO No_Key DO
BEGIN
X := Round( 80*(Key_Loc[I]+1)/(LShaft+2) ) - 1;
YA := 25 - Round(25*(Middle+BORE/2)/(10*BORE));
GotoXY( X,YA ); Write('KEY');
END; {for}
END; {if}
IF No_Hole <> 0 THEN
BEGIN
FOR I := 1 TO No_Hole DO
BEGIN
DrawLine(Hole_Loc[I],Middle+BORE,Hole_Loc[I],-
Middle-BORE/2);
X := Round( 80*(Hole_Loc[I]+1)/(LShaft+2) );
YA := 25 - Round(25*(Middle+BORE)/(10*BORE));
GotoXY( X,YA ); Write('H');
END; {for}
END; {if}
FOR I := 1 TO No_Load DO
BEGIN
X := Round( 80*(Dist_Load[I]+1)/(LShaft+2) ) - 1;
YA := 25 - Round(25*Middle/(10*BORE));
GotoXY( X,YA ); Write('HUB');
END; {for}
GotoXY( 2,13 ); Write('B -- DENOTES BEARING LOCATIONS');
GotoXY( 2,14 ); Write('H -- DENOTES HOLE LOCATIONS');
GotoXY( 2,15 ); Write('GRV -- DENOTES GROOVE LOCATIONS');
GotoXY( 2,17 ); Write('ENTER THE NUMBER OF THE THICK
SECTION >> '); Readln( Sect_Th );
GotoXY( 2,19 ); Write('ENTER THE RADIUS OF THE FILLETS >> ');
Readln( Fillet );
Str_Raise := 2;
IF (Dist_Bear[1] <> 0) AND (Sect_Th = 1) THEN Str_Raise :=
1;
IF (Dist_Bear[1] <> 0) AND (Sect_Th > No_Bear) THEN
Str_Raise := 1;
IF (Dist_Bear[1] = 0 ) AND (Sect_Th = No_Bear) THEN
Str_Raise := 1;
Step1 := 0;
Step2 := 0;
IF (Str_Raise = 2) AND (Dist_Bear[1] = 0) THEN
BEGIN
Step1 := Dist_Bear[Sect_Th] + WB/2;
Step2 := Dist_Bear[Sect_Th+1] - WB/2;
END;
IF (Str_Raise = 2) AND (Dist_Bear[1] <> 0) THEN
BEGIN
Step1 := Dist_Bear[Sect_Th-1] + WB/2;
Step2 := Dist_Bear[Sect_Th] - WB/2;
PROCEDURE Steps1;

LABEL 1;

VAR
  I : Integer;
  Sum : Real;

BEGIN
  Assign( Catalog,Filename );
  Reset( Catalog );
  WHILE NOT EOF( Catalog ) DO BEGIN
    Readln(Catalog,VND,Prt_Num,BORE,OD,WB,FR,DS,DH,CR);
    IF ( CR > CRD ) AND ( BORE >= Dia[1] + 0.125 ) THEN GOTO 1;
  END;
  1: Close( Catalog );

  NumH := 0;
  Sum := 0;
  IF Sect_Th = 1 THEN BEGIN
    Step1 := Dist_Bear[1] - WB/2.0;
    FOR I := 1 TO No_Load DO BEGIN
      IF Dist_Load[I] < Step1 THEN NumH := NumH + 1;
    END;
  END;
  IF Sect_Th >= No_Bear THEN BEGIN
    Step1 := Dist_Bear[No_Bear] + WB/2.0;
    FOR I := 1 TO No_Load DO BEGIN
      IF Dist_Load[I] > Step1 THEN NumH := NumH + 1;
    END;
  END;

  IF NumH > 1 THEN BEGIN
    Writeln;
    Writeln(Bl:10,'**** THIS CASE EXCEEDS THE PROGRAM CAPABILITIES ****');
    Writeln;
    Writeln(Bl:24,'PROGRAM TERMINATED');
Halt;
END; {if}

IF NumH = 0 THEN
BEGIN
  Writeln;
  Writeln('THERE ARE NO HUBS WITHIN THIS SECTION -- NO STEPS REQUIRED ?');
END;

IF ( NumH = 1 ) AND ( Sect_Th = 1 ) THEN
BEGIN
  Dia[1] := DS;
  Sec_Len[2] := Width_Hub[1];
  Dia[2] := DS + 0.125;
  Sec_Len[3] := Step1/10;
  Dia[3] := DS + 0.25;
  Sec_Len[4] := Step1 - Sum;
  Sec_Len[6] := LShaft - ( Step1 + WB );
  Dia[6] := BORE - 0.125;
  No_SecChDia := 6;
END; {if}

IF ( NumH = 1 ) AND ( Sect_Th >= No_Bear ) THEN
BEGIN
  Sec_Len[1] := Step1 - WB;
  Dia[1] := BORE - 0.125;
  Dia[2] := BORE;
  X := ( LShaft - Step1 )/10;
  Sec_Len[3] := ( Dist_Load[No_Load] - Width_Hub[No_Load]/2 ) - X - Step1;
  Dia[3] := DS;
  Sec_Len[4] := ( LShaft - Step1 )/10;
  Dia[4] := DS + 0.25;
  Sec_Len[5] := Width_Hub[No_Load];
  Dia[5] := DS + 0.125;
  Sec_Len[6] := LShaft - ( Dist_Load[No_load] + Width_Hub[No_Load]/2 );
  No_SecChDia := 6;
END; {if}

PROCEDURE Steps2;

VAR
I : Integer;
Sum : Real;

BEGIN
NumH:= 0;
Sum:=0;
WhereToStart:= 0;
FOR I:= 1 TO No_Load DO
  BEGIN
    IF ( Dist_Load[I] > Stepl ) AND ( Dist_Load[I] < Step2 ) THEN
      BEGIN
        NumH:= NumH + 1;
        Point:= I;
        END; {if}
  END; {for}

IF NumH = 0 THEN
  BEGIN
    Writeln;
    Writeln('THERE ARE NO HUBS WITHIN THIS SECTION -- NO STEPS REQUIRED ?');
  END;

IF NumH > 3 THEN
  BEGIN
    Writeln;
    Writeln('**** THIS CASE EXCEEDS THE PROGRAM CAPABILITIES ****');
    Writeln;
    Writeln('PROGRAM TERMINATED');
    Halt;
  END; {if}

IF NumH = 1 THEN
  BEGIN
    WhereToStart:= Point;
    K:= WhereToStart;
    Sec_Len[1]:= Stepl - WB;
    Dia[1]:= BORE - 0.125;
    Sec_Len[2]:= WB;
    Dia[2]:= BORE;
    Sec_Len[3]:= ( Dist_Load[K] - Width_Hub[K]/2 ) - Step1;
    Dia[3]:= DS;
    Sec_Len[4]:= Width_Hub[K];
    Dia[4]:= DS + 0.125;
    Sec_Len[5]:= (Step2 - Stepl)/10;
    Dia[5]:= DS + 0.25;
    Sec_Len[6]:= Step2 - Sum;
Sec_Len[7] := WB;
Dia[7] := BORE;
Sec_Len[8] := LShaft - (Step2 + WB);
Dia[8] := BORE - 0.125;
No_SecChDia := 8;
END; (if)

IF NumH = 2 THEN
BEGIN
WhereToStart := (Point - NumH) + 1;
K := WhereToStart;
Sec_Len[1] := Step1 - WB;
Dia[1] := BORE - 0.125;
Dia[2] := BORE;
Sec_Len[3] := (Dist_Load[K] - Width_Hub[K]/2) - Step1;
Dia[3] := DS;
Sec_Len[4] := Width_Hub[K];
Dia[4] := DS + 0.125;
Sec_Len[5] := (Dist_Load[K+1] - Width_Hub[K+1]/2) - Sum;
Dia[5] := DS + 0.25;
Sec_Len[6] := Width_Hub[K+1];
Dia[6] := DS + 0.125;
Sum := Sum + Sec_Len[5] + Sec_Len[6];
Sec_Len[7] := Step2 - Sum;
Dia[7] := DS;
Sec_Len[8] := WB;
Dia[8] := BORE;
Sec_Len[9] := LShaft - (Step2 + WB);
Dia[9] := BORE - 0.125;
No_SecChDia := 9;
END; (if)

IF NumH = 3 THEN
BEGIN
WhereToStart := (Point - NumH) + 1;
K := WhereToStart;
Sec_Len[1] := Step1 - WB;
Dia[1] := BORE - 0.125;
Dia[2] := BORE;
Sec_Len[3] := (Dist_Load[K] - Width_Hub[K]/2) - Step1;
Dia[3] := DS;
Sec_Len[4] := Width_Hub[K];
Dia[4] := DS + 0.125;
Sec_Len[4];
Sec_Len[5] := (Dist_Load[K+1] -
Width_Hub[K+1]/2) - Sum;
Dia[5] := DS + 0.25;
Sec_Len[6] := Width_Hub[K+1];
Dia[6] := DS + 0.375;
Sum := Sum + Sec_Len[5] + Sec_Len[6];
Sec_Len[7] := (Dist_Load[K+2] -
Width_Hub[K+2]/2) - Sum;
Dia[7] := DS + 0.5;
Sec_Len[8] := Width_Hub[K+2];
Dia[8] := DS + 0.25;
Sum := Sum + Sec_Len[7] + Sec_Len[8];
Sec_Len[9] := Step2 - Sum;
Dia[9] := DS;
Sec_Len[10] := WB;
Dia[10] := BORE;
Sec_Len[11] := LShaft - (Step2 + WB);
Dia[11] := BORE - 0.125;
No_SecChDia := 11;
END; {if}
END; {procedure steps2}

PROCEDURE Min_MaxMEI;

VAR
  I : Integer;

BEGIN
  Temp1 := 100000.0;
  Temp2 := -100000.0;
  FOR I := a TO MX-1 DO
  BEGIN
    IF Mom_El[I] >= Temp2 THEN
    BEGIN
      Max_Y := Mom_El[I];
      Temp2 := Max_Y;
    END;
    IF Mom_El[I] <= Temp1 THEN
    BEGIN
      Min_Y := Mom_El[I];
      Temp1 := Min_Y;
    END;
  END; {for}
END; {procedure Min_MaxMEI}

PROCEDURE MomEI_Dia;

VAR
  ch : Char;
  I, J, IX, IY, Flag : Integer;
BEGIN
EnterGraphic;
GotoXY(1,1);
Writeln(' [MOMENT*1000]/[E*I] DIAGRAM');
Writeln;
Writeln('DO YOU WANT A HARDCOPY OF THE DIAGRAM Y/N ?');
Write('ANSWER » ');
Read( ch ); ch:= UpCase( ch );
IF ch = 'Y' THEN Flag:= 1 ELSE Flag:= 0;
ClearScreen;
Bottom:= Abs(0.125*(Max_Y-Min_Y));
Top:= Abs(0.125*(Max_Y-Min_Y));

DefineWorld(3,-1.1,Min_Y-Bottom,LShaft+1.2,Max_Y+Top);
SelectWorld(3);
SelectWindow(1);

DrawSquare(-1.1,Min_Y-Bottom,LShaft+1.2,Max_Y+Top,false);

SetLineStyle(1);
DrawLine(0,Min_Y,LShaft,Min_Y);
DrawLine(0,Min_Y,0,Max_Y);
DrawLine(0,0,LShaft,0);

Tick:= LShaft/10;
Range:= LShaft+2.3;
Sfx:= 80/Range;
XX:= Abs(0.25*Bottom);
SetLineStyle(0);

FOR I:=1 TO 10 DO
BEGIN
Mark:= Tick*I;
DrawLine( Mark, Min_Y, Mark, Min_Y+XX );
X:= 1 + Tick*I;
IX:= Round( Sfx*X );
GotoXY( IX,24 ); Write( Mark:4:1 );
END; {for}

Tick:= (Max_Y-Min_Y)/10;

FOR I:= 1 TO 10 DO
BEGIN
Mark:= Min_Y + Tick*I;
DrawLine( 0, Mark, 0.02*LShaft, Mark );
END; {for}

Range:= (Max_Y + Top) - (Min_Y - Bottom);
Sfy:= 25/Range;
Bottom:= Bottom - 0.125*Bottom;

FOR I:= 1 TO 10 DO
BEGIN
  Y:= Bottom + I*Tick;
  IY:= 25 - Round( Y*Sfy );
  Mark:= Min_Y + I*Tick;
  GotoXY( 1,IY ); Write( Mark:6:3 );
END;

Temp1:= 0;
Temp2:= 0;
FOR I:= 0 TO MX-1 DO
BEGIN
  DrawLine( Temp1, Temp2, I*DX, Mom_EI[I] );
  Temp1:= I*DX;
  Temp2:= Mom_EI[I];
END; {for}

GotoXY( 73,9); Write('M*1000');
GotoXY( 75,10); Write('----');
GotoXY( 76,11); Write('EI');

IF Flag = 1 THEN
BEGIN
  SelectScreen(1);
  HardCopy( false,1 );
END; {if}

Repeat
  Read( Kbd, ch )
Until ch = ' '
LeaveGraphic;
END; {procedure MomEI_Dia}

BEGIN
  Change_PROP:= 0;
  Writeln;
  REPEAT
  Writeln(' DO YOU WANT ONLY TO CHANGE THE ULTIMATE
AND/OR TENSILE?');
  Writeln('STRENGTH OF THE MATERIAL?');
  Write('ANSWER Y OR N >> ');
  Readln( Opt ); Opt:= UpCase( Opt );
  IF Opt = 'Y' THEN
  BEGIN
    Writeln;
    Write('ENTER THE YIELD STRENGTH (psi) >> ');
    Readln(S_Ys);
    Writeln;
  END;
Write('ENTER THE ULTIMATE TENSILE STRENGTH (psi) >> ');  
Readln(S_Ult);  
Change_Prop:= 1;  
END; {if}

IF Opt = 'Y' THEN GOTO 2;

IF Opt = 'N' THEN
BEGIN
  REPEAT
    Writeln('DO YOU WANT TO CHANGE THE MATERIAL OR PROPERTIES ?');
    Write('ANSWER Y OR N >> ');  
    Readln(ans); ans:= UpCase(ans);
    IF ans = 'Y' THEN
      BEGIN
        Writeln;
        REPEAT
          Writeln('MATERIAL 1. CARBON STEEL 2. ALUMINUM');
          Write('CHOOSE A NUMBER >> I');  
          Readln(Choice);
          CASE Choice OF
            '1': Mod_Elast:= 3.0E+07;
            '2': Mod_Elast:= 1.0E+07;
            ELSE Writeln('ERROR: INVALID SELECTION, PLEASE TRY AGAIN.');
          END; {case}
        UNTIL ( Choice = '1' ) OR ( Choice = '2' );
        Writeln;
        Write('ENTER THE YIELD STRENGTH (psi) >> ');  
        Readln(S_Ys);
        Writeln;
        Write('ENTER THE ULTIMATE TENSILE STRENGTH (psi) >> ');
        Readln(S_Ult);
      END; {if}
    IF ( ans <> 'Y' ) AND ( ans <> 'N' ) THEN Writeln('ERROR : INVALID SELECTION -- ENTER Y OR N');
    UNTIL ( ans = 'Y' ) OR ( ans = 'N' );
  END; {if}
IF ( Opt <> 'Y' ) AND ( Opt <> 'N' ) THEN Writeln('ERROR : INVALID RESPONSE -- ENTER Y OR N');
UNTIL ( Opt = 'Y' ) OR ( Opt = 'N' );
ClrScr;
Writeln;
REPEAT
  Writeln('DO YOU WANT TO CHANGE SHAFT LENGTH OR BEARING INFORMATION ?');
  Write('ANSWER Y OR N >> ');}
Readln( ans ); ans:= UpCase( ans );
IF ans = 'Y' THEN
BEGIN
  Writeln;
  Write('ENTER THE LENGTH (in.) OF THE SHAFT >> ');
  Readln(LShaft);
  Writeln;
  Repeat
  Write('ENTER THE NUMBER OF BEARINGS >> ');
  Readln(No_Bear);
  IF No_Bear <= 1 THEN Writeln('ONE BEARING YIELDS AN UNSTABLE PROBLEM -- IMPOSSIBLE SOLUTION');
  Until No_Bear > 1;
  FOR I:= 1 TO No_Bear DO
  BEGIN
    Writeln;
    Write('ENTER THE DISTANCE(in.) TO BEARING( ',I,' ) >> ');
    Readln( Dist_Bear[I] );
  END;
END; {if}
ELSE
BEGIN
  Writeln('ERROR: INVALID RESPONSE -- ENTER Y OR N');
  Until ( ans = 'Y' ) OR ( ans = 'N');
END; {if}

Design_Area;

FOR I:= 1 TO 11 DO
BEGIN
  Dia[I]:= 0.0;
  Mom[I][I]:= 0.0;
  Sec_Len[I]:= 0.0;
END;

FOR I:= 1 TO 10 DO
BEGIN
  Grv_Loc[I]:= 0.0;
  Key_Loc[I]:= 0.0;
  Hole_Loc[I]:= 0.0;
END; {for}

FOR I:= 0 TO MX-1 DO
BEGIN
  Torque[I]:= 0.0;
  D[I]:= 0.0;
  Moment[I]:= 0.0;
  Shear[I]:= 0.0;
  Mom_EI[I]:= 0.0;
END; {for}

X:= LShaft/20;
\[ Y := \text{Round}(X \times 16); \]
\[ \text{Dia}[1] := \frac{Y}{16}; \]
\[ \text{MomI}[1] := \frac{(\pi \times \text{Dia}[1] \times \text{Dia}[1] \times \text{Dia}[1] \times \text{Dia}[1])}{64}; \]

\[ \text{No}_\text{Seg} := 50; \]
\[ \text{DX} := \frac{\text{LShaft}}{\text{No}_\text{Seg}}; \]
\[ \text{Delta}_\text{Conc} := \text{DX} \times \text{DX} \times \text{DX}; \]

\textbf{Repeat}
\textbf{WriteLn};
\textbf{WriteLn}('DO YOU WANT TO CHANGE THE LOADING INFORMATION?');
\textbf{Write}('ANSWER Y OR N >> ');
\textbf{Readln( ans ); ans := UpCase( ans );}
\textbf{IF ans = 'Y' THEN}
\textbf{BEGIN}
\textbf{WriteLn};
\textbf{Write}('ENTER THE NUMBER OF HUBS >> ');
\textbf{Readln( No_Load );}
\textbf{FOR I:= 1 TO No_Load DO}
\textbf{BEGIN}
\textbf{WriteLn};
\textbf{Write}('ENTER THE DISTANCE (in.) TO HUB(',I,') >> ');
\textbf{Readln( Dist_Load[I] );}
\textbf{Write}('ENTER THE WIDTH (in.) OF HUB(',I,') >> ');
\textbf{Readln( Width_Hub[I] );}
\textbf{Write}('ENTER THE CONC. LOAD(lbs) ON HUB(',I,') >> ');
\textbf{Readln( Load[I] );}
\textbf{CLoad[I] := Load[I];}
\textbf{Load_Loc[I] := \text{Round}( Dist_Load[I]/\text{DX} ) + 1;}
\textbf{CLoad_Loc[I] := Load_Loc[I];}
\textbf{Write}('ENTER THE TORQUE(in-lbs) ON HUB(',I,') >> ');
\textbf{Readln( Torq[I] );}
\textbf{Write}('ENTER THE THRUST(lbs) ON HUB(',I,') >> ');
\textbf{Readln( Thrust[I] );}
\textbf{END; \{ for \}}
\textbf{END; \{ if \}}
\textbf{IF ans = 'N' THEN}
\textbf{BEGIN}
\textbf{FOR I:= 1 TO No_Load DO}
\textbf{BEGIN}
\textbf{Load[I] := CLoad[I];}
\textbf{Load_Loc[I] := \text{Round}( Dist_Load[I]/\text{DX} ) + 1;}
\textbf{CLoad_Loc[I] := Load_Loc[I];}
\textbf{END; \{ for \}}
\textbf{END; \{ if \}}
IF ( ans <> 'Y' ) AND ( ans <> 'N' ) THEN Writeln('ERROR :
INVALID RESPONSE -- ENTER Y OR N ?');
Until ( ans = 'Y' ) OR ( ans = 'N' );

TFx := 0.0;
FOR I := 1 TO No_Load DO
  TFx := TFx + Thrust[I];

Axial_Thr[1] := 0.0;
Axial_Thr[2] := 0.0;
IF TFx > 0 THEN Compr_End := Round(Dist_Bear[No_Bear]/DX)
ELSE Compr_End := Round(Dist_Bear[1]/DX);

IF No_Load > 1 THEN
  { FINDS THE TORQUE
  DISTRIBUTION IN THE SHAFT }

IF No_Bear > 0 THEN
  BEGIN
    FOR I := 1 TO No_Bear DO
      {find bearing
       matrix position}
      Bear_Pos[I] := Round ( Dist_Bear[I]/DX );
  END; {if}

MatSize := Trunc( LShaft/DX - (No_Bear - 1) );

IF Bear_Pos[1] = 1 THEN MatSize := MatSize-1;
IF Bear_Pos[No_Bear] = Trunc( (LShaft-DX)/DX ) THEN
  MatSize := MatSize-1;

Writeln; Writeln('PERFORMING CALCULATIONS --- PLEASE WAIT.');
Writeln; Writeln('STARTING DIAMETER = ',Dia[1]:6:4);
Writeln;

GetMem( Stiff, Sizeof( Matrix ) );
K := 0;
1: In_Matrix;

Over_Write;

Inertia;

Invert;
Temp := -1.0;
FOR I := 1 TO MatSize DO
  IF Abs( Stiff^[I,MatSize+1] ) > Temp THEN Temp := Abs( Stiff^[I,MatSize+1] );
IF K < 6 THEN
  BEGIN
    Writeln(' MAX. DEFLECTION = ',Temp:7:5);
IF Temp > 0.01 THEN
BEGIN
  Writeln('Diameter 1 = ', Dia[1]:5:4);
  IF Temp > 0.09 THEN Dia[1] := Dia[1] + 0.5;
  IF ( Temp <= 0.09 ) AND ( Temp > 0.04 ) THEN Dia[1] := Dia[1] + 0.1875;
  IF ( Temp <= 0.04 ) AND ( Temp > 0.014 ) THEN Dia[1] := Dia[1] + 0.125;
  IF ( Temp <= 0.014 ) AND Temp > 0.01 THEN Dia[1] := Dia[1] + 0.0625;
  Writeln('Increase Diameter = ', Dia[1]:5:4);
  MomI[1] := (PI*Dia[1]*Dia[1]*Dia[1]*Dia[1])/64;
  FOR J := 1 TO No_Load DO BEGIN
    Load[J] := CLoad[J];
    Load_Loc[J] := CLoad_Loc[J];
  END;
  Writeln('DEFLECTION OF SHAFT EXCEEDS 0.01');
  K := K + 1;
  Writeln; Writeln(B1:22, 'ITERATION ', K);
  GOTO 1;
END; {if}
END; {if}
IF K = 6 THEN Writeln('DEFLECTIONS MAY EXCEED 0.01 BUT CONTINUING PROGRAM EXECUTION');
Copy_Deflect;
FreeMem( Stiff, Sizeof( Matrix ) );
Beep;
VShear;
TMoment;
Max_MinSH;
SH_Dia;
Min_MaxMom;
Mom_Dia;
Repeat
  Writeln;
  Writeln('DO YOU WANT A HARDCOPY OF THE LOAD AND STRESS DISTRIBUT. ON SHAFT ?');
  Write('ANSWER Y OR N >> ');
  Readln( ans ); ans := UpCase( ans );
  IF ans = 'Y' THEN HDCopy;
  Until ( ans = 'Y' ) OR ( ans = 'N' );
ClrScr;
Writeln('*** BEARING SELECTION ***');
REPEAT
  Writeln;
  Write('WHICH RACE ROTATES (I)NNER OR (O)UTER >> ');
  Readln( Choice ); Choice:= UpCase( Choice );
  CASE Choice OF
    'I' : V:= 1.0;
    'O' : V:= 1.2;
  ELSE Writeln('ERROR: INVALID SELECTION, PLEASE TRY AGAIN');
  END
  UNTIL ( Choice = 'I' ) OR ( Choice = 'O' );
Writeln;
Write('ENTER THE SPEED OF ROTATION (RPM) >> ');
Readln( NSpeed );
Write('ENTER THE DESIGN LIFE (HOURS) >> ');
Readln( DHours );
Write('ENTER THE BEARING RELIABILITY >> ');
Readln( BRel' );
Writeln;
Write('ENTER THE BEARING CATALOG FILENAME >> ');
Readln( Filename );
FOR I:= 1 TO Length( Filename ) DO
  Filename[I]:= UpCase( Filename[I] );
  Filename:= Concat( Filename,Ftype );
FOR I:= 1 TO No_Bear DO
  BEGIN
    Writeln;
    REPEAT
      Write('IS BEARING(',I,'} (R)OLLER OR (B)ALL ? >> ');
      Readln( Choice ); Choice:= UpCase( Choice );
      CASE Choice OF
        'R' : Bear_Type[I]:= 0;
        'B' : Bear_Type[I]:= 1;
      ELSE Writeln('ERROR: INVALID SELECTION, PLEASE TRY AGAIN');
      END
      UNTIL ( Choice = 'R' ) OR ( Choice = 'B' );
    END; {for}
Bear_Select;
ClrScr;
Writeln;
Writeln('THE FOLLOWING QUESTIONS ARE FOR THE TENATIVE LOCATIONS OF');
Writeln(' HOLES, GROOVES, AND KEWAYS');
Writeln;
Write('ENTER THE NUMBER OF HOLES >> ');
Readln( No_Hole );
IF No_Hole <> 0 THEN
BEGIN
  FOR I := 1 TO No_Hole DO
  BEGIN
    Writeln;
    Write('ENTER THE LOCATION OF HOLE(',I,') >> ');
    Readln( Hole_Loc[I] );
  END {for}
END; {if}
Writeln;
Write('ENTER THE NUMBER OF KEYWAYS >> ');
Readln( No_Key );
IF No_Key <> 0 THEN
BEGIN
  FOR I := 1 TO No_Key DO
  BEGIN
    Writeln;
    Write('ENTER THE LOCATION OF KEYWAY(',I,') >> ');
    Readln( Key_Loc[I] );
  END {for}
END; {if}
Writeln;
Write('ENTER THE NUMBER OF GROOVES >> ');
Readln( No_Grv );
IF No_Grv <> 0 THEN
BEGIN
  FOR I := 1 TO No_Grv DO
  BEGIN
    Writeln;
    Write('ENTER THE LOCATION OF GROOVE(',I,') >> ');
    Readln( Grv_Loc[I] );
  END {for}
END; {if}
Writeln;
Writeln('PRESS THE SPACE BAR TO CONTINUE ...');

Repeat
  Read( Kbd,ans )
Until ans = ' ';

Thick_Section;

IF Str_Raise = 1 THEN Steps1;
IF Str_Raise = 2 THEN Steps2;

IF NumH <> 0 THEN
BEGIN
  FOR I := 1 TO No_SecChDia DO
    MomI[I] := (PI*Dia[I]*Dia[I]*Dia[I]*Dia[I])/64;
BEGIN
J:= 1;
Sum:= Sec_Len[J];
FOR I:= 0 TO MX-1 DO
BEGIN
  IF J < No_SecChDia THEN
    BEGIN
      IF (I*DX) > Sum THEN
        BEGIN
          J:= J + 1;
          Sum:= Sum + Sec_Len[J];
        END;
    END;
  END;
Mom_EI[I]:= 1000*Moment[I]/(Mod_Elast*MomI[J]);
END; {for}
END; {if}

Min_MaxMEI;

MomEI_Dia;

2: Assign(ChainFile,'SubProgm.chn');
   Chain(ChainFile);
END. {program}