EPRO1004 - 202002
Home-based Programming Assignment

Instructions (Read Carefully):

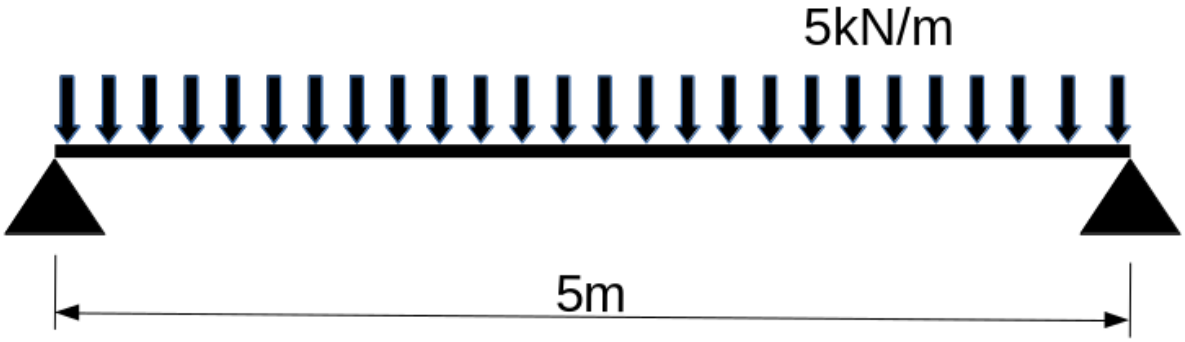
1. This is an open book programming assignment. You are allowed to use online resources to assist in this assignment;
2. This is an individual assignment and no collusion with other students of any kind is allowed;
3. Your program has to work. That means it has to compile correctly, run according to specifications, and give correct results. The submitted assignment must also meet the following criteria:
    - Good design, including good **pseudo code, logic diagrams** and algorithms
4. You must observe good programming practices, including comment, documentation, and readability of your program;
5. To facilitate program readability, you must include function calls in your program whenever possible;
6. All names of constants, types, variables, functions, etc. must be "self-documenting" and "self-describing." For example, you should name a variable ***total*** if it is used to store a total. Give a function a name like ***printList*** if its job is to print a list;
7. This home-based programming assignment constitutes **40% of the overall final mark of the EPRO1004 unit**.
8. **Final submission deadline is 23:00, 20th September. The instructions on how to submit your file will be provided by your Unit Coordinator as soon as possible.**

Calculating the ratio of applied load to capacity of a structural steel beam under varying conditions

Background

The design of a steel beam must take into account a wide range of factors. Material strength, load position, magnitude of load, end restraints and compressive flange restraint are just some of the factors that must be taken into account.

This assignment will involve writing a simple application that will calculate the bending capacity of a simply supported steel beam, and compare it to the maximum moment brought about via a full length distributed load. A range of beam lengths and applied loads will be used in calculations and the results displayed in a table. A sample length/loading situation is illustrate below.



Two main equations govern the design of beams for bending. The first covers what is called the section capacity, M_s, the second the member capacity, M_b. In turn these equations are compared to the applied maximum bending moment:

$$M^* \leq \phi M_s \qquad\qquad M^* \leq \phi M_b$$

where ϕ = 0.9, M^* is the applied moment

The calculation of M_s is quite straight forward:

$$M_s = f_y Z_e$$

where:

M_s is the section bending capacity

f_y is the yield stress – in this exercise this is taken to be 300MPa

Z_x is the effective section modulus – this is read from provided section tables (mm^3)

The calculation of M_b is more complex, but still relatively straight forward.

$$M_b = \alpha_m \alpha_s M_s \leq M_s$$

where:

M_b is the member bending capacity (Nmm)

α_m = Moment modification factor – in this exercise this is taken as 1

α_s = Slenderness reduction factor

$$\alpha_s = 0.6\left[\sqrt{\left[\left(\frac{M_s}{M_o}\right)^2 + 3\right]} - \left(\frac{M_s}{M_o}\right)\right]$$

M_o is the reference elastic buckling moment for a member subject to bending (Nmm)

$$M_o = \sqrt{\left(\frac{\pi^2 EI_y}{l_e^2}\right)\left[GJ + \left(\frac{\pi^2 EI_w}{l_e^2}\right)\right]}$$

E = Modulus of elasticity – in this exercise this is taken as 200,000MPa

I_y = The second moment of area about the minor y-axis - this is read from provided section tables (mm^4)

G = Shear Modulus – in this exercise this is taken as 80,000MPa

J = Section torsion constant – this is read from provided section tables (mm^3)

I_w = Section warping constant – this is read from provided section tables (mm^4)

l_e = Effective length – provided by the application (mm)

The applied moment, M^*, will be calculated by the application using a value for an applied *Uniformly Distributed Load*, UDL. The application will use the simple equation:

$$M^* = \frac{wl_e^2}{8}$$

where:

M^* is the maximum moment at mid-span in kNm

w = Applied full length UDL in kN/m

l = The length of the beam – this will be equivalent to l_e (m)

It is important to note that the value for length in the capacity calculations is in ***mm*** whereas when calculating the applied moment, the length is in ***m***. So for a length of ***3m***, the capacities will be calculated using ***3000mm*** but for the applied load, it will be calculated using ***3m***. In turn this results in M_s and M_b being calculated in ***Nmm*** whereas M^* is calculated in ***kNm.*** It is recommended that for the calculation of the capacity ratio that the capacities be divided by 10^6 to bring them into ***kNm***.

The ratio that is calculated is derived by dividing the applied moment by the two capacities and in turn taking the larger:

$$ratio = MAX\left(\frac{M^*}{M_s}, \frac{M^*}{M_b}\right)$$

Requirements

At its lowest level the app will take an effective length and in conjunction with member parameters calculate the section and member bending capacity of the member for said length. It will, for the same length, calculate a bending moment based upon an applied UDL. The bending moment will then be divided by the section & member bending capacities respectively to provide ratios. The larger ratio will then be displayed to the user.

```
 ** Beam Capacity Ratio Calculator **

For a 410UB82 the ratio of applied UDL to
moment capacity is as follows:

 UDL                          Effective Length (m)
(kN/m)*      0      1      2      3      4      5      6      7      8      9     10
*************************************************************************************
    1 *   0.9700 0.6300 0.0300 0.9400 0.5600 0.7700 0.2200 0.0500 0.6400 0.3600 0.1200
    2 *   0.9900 0.1800 0.8300 0.5800 0.9900 0.0800 0.4000 0.2300 0.5600 0.8600 0.2300
    3 *   0.2100 0.2100 0.9200 0.7000 0.7100 0.0900 0.6100 0.1300 0.2600 0.5900 0.2800
    4 *   0.2900 0.0500 0.8400 0.5800 0.7900 0.4100 0.2300 0.1500 0.5300 0.7400 0.8600
    5 *   0.8900 0.3200 0.8500 0.9700 0.7200 0.0800 0.0500 0.1000 0.8400 0.2600 0.3200
    6 *   0.7600 0.4900 0.0300 0.3800 0.1000 0.1600 0.6400 0.2100 0.4400 0.9300 0.7800
    7 *   0.8000 0.0300 0.5700 0.7300 0.2600 0.2500 0.7900 0.0000 0.1100 0.6800 0.8400
    8 *   0.4800 0.1700 0.0800 0.0800 0.2300 0.1900 0.9200 0.0100 0.5100 0.6900 0.0200
    9 *   0.5400 0.0700 0.1300 0.2300 0.7100 0.8600 0.1900 0.1600 0.6500 0.5200 0.1900
   10 *   0.2200 0.2500 0.9800 0.9900 0.0400 0.5000 0.1000 0.2400 0.3500 0.1000 0.4200
```

Figure 1: Example App Output. n.b. Values are for illustration only

As can be seen in Figure 1, the app will generate UDL values from 1kN/m to 10 kN/m with 1kN/m increments. In turn, for each load value, bending moments and bending capacities will be calculated for effective lengths ranging from 0m to 10m with 1m increments. Ratios of bending moment to capacity, both section and member, are to be calculated for each UDL – effective length pair. The result, the greater of the two ratios, will be displayed in a table, similar to Figure 1.

Note: An effective length of 0m is a special case which relates to what is known as *full lateral restraint*. An explanation of this is well outside the scope of this subject. You will only have to calculate the section capacity but you will need to calculate an applied moment for each UDL for an assumed effective length of 5m.

To control the app, the user will use a simple text file. The text file will contain the name of the steel section being investigated and the parameters that are listed in the Background section above as being *read from provided section tables*. Using a ***460 UB 82*** as an example, a possible input file format would be:

```
460UB82
 1610000
 18600000
 701000
 919000000000
```

The data is sourced from a pdf file that is available on Moodle, Week 8, titled ***Steel Section Properties***.

The input file can have any name, but the app must prompt the user for this name. Once entered the app will open it and read its contents. Note that there should be no spaces in the beams name string. Once the data file has been read in, the calculations will take place.

Output in a format similar to Figure 1 is expected to both the screen and to a file that is named after the steel section being investigated. The above section would provide output to a file called *460UB82.dat.*

Programming Tasks

Your task is to develop a working C program that will:

- Prompt the user for the name of a data file that contains the data needed to calculate the section and member bending capacities.
- Open then read the data file. If the file is not found, appropriate action should be taken.
- Calculate the bending moment and capacities for each combination of UDL and effective length. Calculate the appropriate ratios.
- Output the results to both the screen and file

You should demonstrate good programming practise, making use of functions and return values. As files are being opened, care should be taken to ensure the opening has actually succeeded. *Magic Numbers* should not be used. The code should be well laid out and consistent in format. Variable names are expected to provide an indication of their purpose.

What you have to submit

1. The *pseudo code* and flow charts documenting the design of the program. This should be submitted as a pdf named **design.pdf** (20%);
2. The testing document. This will provide hand calculations to show that the calculations in the app are correct. It should also indicate what happens when erroneous data is entered. For example what happens if not all of the data file is present or a value is negative. This will be called **testing.pdf** (20%);
3. Well-documented, syntax-free and correctly-compiling C source code. This should include an example data file. Name this file **program.c** (50%);
4. A compliance/summary document. This is the developer showing that the app produces results that match with the testing document. Also if there are any conditions when the app is run. This will be called **comply.pdf** (10%).

Submission Guideline:

Submission will be through Moodle. Submission will be in three parts on three different dates.

Item	Date
Design document, design.pdf	23:00, Sunday 30th of August
Testing Document, testing.pdf	23:00, Sunday 6th of September
Code & compliance, program.c & comply.pdf	23:00, Friday 18th of September

More information will be provided closer to the day.

Appendix

Marking Rubric for Programming

Criteria	**Exceptional** (>=80%)	**Acceptable** (between 65% and 79%)	**Amateur** (between 50% and 64%)	**Unsatisfactory** (<50%)
Specifications	The program works and meets all of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
Readability	The code is exceptionally well organized and very easy to follow.	The code is fairly easy to read.	The code is readable only by someone who knows what it is supposed to be doing.	The code is poorly organized and very difficult to read
Reusability	The code could be reused as a whole or each routine could be reused.	Most of the code could be reused in other programs.	Some parts of the code could be reused in other programs.	The code is not organized for reusability.
Documentation	The documentation is well written and clearly explains what the code is accomplishing and how.	The documentation consists of embedded comment and some simple header documentation that is somewhat useful in understanding the code.	The documentation is simply comments embedded in the code with some simple header comments separating routines.	The documentation is simply comments embedded in the code and does not help the reader understand the code.
Delivery	The program was delivered on time.	The program was delivered within a week of the due date.	The code was within 2 weeks of the due date.	The code was more than 2 weeks overdue.
Efficiency	The code is extremely efficient without sacrificing readability and understanding.	The code is fairly efficient without sacrificing readability and understanding.	The code is brute force and unnecessarily long.	The code is huge and appears to be patched together.

Source

https://uwf.edu/media/university-of-west-florida/academic-affairs/departments/cutla/documents/Computer-Programming-Grading-Rubric---California-State-University-Long-Beach.pdf

Marking Rubric for Pseudocode design

Criteria	Exceptional (>=80%)	Acceptable (between 65% and 79%)	Amateur (between 50% and 64%)	Unsatisfactory (<50%)
Algorithm	Complete working algorithm created with strong supporting understanding of the domain area.	Incomplete nearly working algorithm created with strong supporting understanding of the domain area.	Incomplete non working algorithm created with incomplete supporting understanding of the domain area.	No working algorithm created or no supporting understanding of the domain area.
Readability	Pseudocode is very clear, understandable, readable, and organised into as few steps necessary as possible.	Pseudocode is clear, understandable, and the code is organised into steps, though there may be a few extra steps.	Pseudocode is somewhat understandable, but very little of the code is organised logically.	Pseudocode is difficult to understand and is unorganised.
Ability to convert to a program	The code can be easily converted into a program.	The code can be converted into a program.	The code can be converted into a program only with much difficulty.	Code cannot be converted into a program.

Source

https://www.rcampus.com/rubricshowc.cfm?sp=yes&code=FX2XX4X&