**ENGR 102 – Fall 2021**
**Lab Assignment #5b**

**Deliverables:**

There are two deliverables for this individual assignment. Please submit the following files to Mimir:

- `Lab5b_Act1.pdf`
- `Lab5b_Act2.py`

Remember to include the header and please **do not** zip the files.

**Activity #1**: *Planning a large program*

In this individual assignment, you should follow a similar strategy to the one pursued for your team assignment. Note: in comparison to the team program, there are far fewer conditions and necessary test cases. However, there is more computation, including practice interpolating data (remember Lab 2a?). You may want to review the process for interpolation. See the document titled "Linear Interpolation" posted in Week 2 on Canvas.

Your task is to create a program that will report what the stress is for a given strain, for structural steel. You will be creating a simplified model for the relationship between stress and strain, and then doing computations with it. Stress-strain relationships are important in several engineering disciplines. In very rough terms, the strain of an object tells us how much it has deformed, and the stress on an object tells us how much force the material is exerting in response. Different materials have different stress-strain relationships. These relationships are often displayed graphically as a stress-strain curve, plotting strain on the x-axis and stress on the y-axis. An example curve is shown below. The image was taken from http://www.ce.memphis.edu/1101/notes/concrete/section_1_strength_of_materials.html, which includes a more complete description. If you perform a web search on "Stress-strain curve for steel", and look at images, you will see several other examples, both idealized and measured. Different materials will have differently-shaped stress-strain curves.

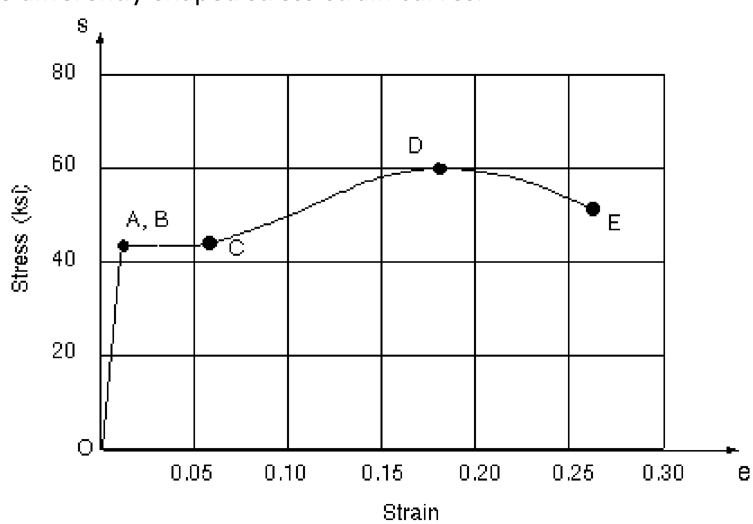


Figure 3. Stress-strain diagram for structural steel in tension.

**Use the following points:** A: (0.01, 43) C: (0.06, 43.5) D: (0.18, 60) E: (0.27, 51)

For this part of the assignment, put together a document that you will use to plan your program. You will submit a PDF of this document titled Lab5b_Act1.pdf. Do this **before** you start to code.

First, examine the stress-strain curve. The curve has an increasing linear elastic region (from O to A) in which the stress is directly proportional to the strain. The slope of this region is called Young's Modulus. After this (point A to point B), there continues to be increasing stress per strain, but not at a constant rate. From the upper yield point (B) to the lower yield point (C) is a "plastic" region that is not flat (although it appears so in the figure above). This is followed by a "strain hardening" region (C to D) up to the point of "maximum strength (D), a "necking" region (D to E) and finally a fracture point (E). In reality, points A and B are distinct, and the plastic region is not purely linear. In your model, you may assume points A and B are the same, and the plastic region is nearly flat.

You are to come up with a simplified, purely linear model of the stress-strain curve. That is, you are to approximate the curve by a series of **straight-line segments** between the marked points. You should approximate the graph using 4 linear segments. The first line should begin at (0,0) and the last ($4^{th}$) line should end at the fracture point. Using more lines will give a more accurate representation, but will be more work in terms of coding. For this assignment, you do not need to be precise, and should use just 4 line segments and the points (A, C, D, and E) given below the graph.

- For the linear elastic portion, state in your pdf document the value of Young's Modulus.
- Record the endpoints (as strain and stress "coordinates") for each of the linear segments.

You'll now work on taking that linear approximation of the stress-strain curve, and creating a program that can evaluate it for you (given a strain, calculate and report the stress).

Next, consider what values you need to store, and the general steps you will need to follow in your program.:

- Make a list of the variables you think you will need, and the names you will use.
- Create a sequence of steps that you will follow
    - If you have a conditional statement (you should have a few!), you might want to indicate each part of the condition as a separate action.
    - The computation will involve a few stages. Please separate the stages into different parts; do not just say "compute stress".

Next, create a list of test cases that you will use in your program. Be sure to handle both "typical" and "edge" cases. **Do this before writing the program itself!**

- Similar to the group activity, for each test case provide the input, the expected output, and identify the type of case ("typical" or "edge").
- Note that you should try to come up with a complete set of test cases that thoroughly test the idea. You should have **at least** 10 test cases.
- Your program should handle input strains outside the provided range on the graph by printing a message to the screen.

******* Your PDF should contain the value of Young's Modulus, your variable list, your sequence of steps, and your test cases. *******

**Activity #2**: *Constructing your program*

**AFTER** doing Activity #1, construct your program and name the file Lab5b_Act2.py. Your program should ask users for a strain, and report the stress. As you write your program, please be sure to do the following:

- Include comments for your program. It's a good idea to begin by converting your list of steps into comments.
- Develop incrementally. Write some code then test it before writing the next section of code.
- Be sure your program runs and passes all of your test cases. You can submit your code as many times as you want to Mimir. This is a good way to check your code and see if it passes the test cases there.
- Your code should output in the format below, using one (1) decimal place for the calculated stress.

✓ *Example output* (using input: **0.05**)

```
Enter the strain: 0.05
The stress is approximately 43.4
```

✓ *Example output* (using input: **-1**)

```
Enter the strain: -1
Strain is undefined in that region
```