# ENGR 102 - Lab Assignment #4B

Fall 2019

You are to write four programs as described below. Remember to document your code with comments, print what the program does to the user, and print labels and units when applicable. When these programs are completed, submit all files to eCampus. Remember the appropriate header information.

---

**Program 1**

☑ *Prepare and code following the input, calculate, output method.*
☑ *Use boolean and conditional statements.*

Write a program to read in a person's height and weight. Report whether the person is underweight, at a healthy weight, overweight or obese.

See: https://www.cdc.gov/healthyweight/assessing/bmi/adult_bmi/index.html

---

**Program 2**

☑ *Prepare and code following the input, calculate, output method, using well-formatted prompts and results.*
☑ *Assign values to variables, and give variables appropriate names for their use.*
☑ *Use boolean and conditional statements.*
☑ *Find and use engineering resource material from reliable sources.*

Reynold Number (Re) is a dimensionless parameter that describes the ratio of inertial forces to viscous forces of a fluid flow. The value of Re indicates whether the flow is laminar, in transition, or turbulent. The exact transition points below which flow is laminar, or above which it is turbulent have varying definitions. Find (and cite) an online source that gives values that you use for these levels.

Write a program to read in values of a fluid velocity, kinematic viscosity and characteristic linear dimension to compute a Reynolds number. Report whether flow is laminar, transient or turbulent.

Please note that you must pay attention to the units of these parameters so that they will cancel each other during the calculation and leave the result dimensionless. Require the user inputs to be in appropriate units so your result is a valid Reynolds number.

*Challenge (optional, for glory only):*

Hard code the viscosity values for three different fluids in your code. Then have the code prompt the user to enter, for example, "water" or "air" or "oil" followed by the velocity, and pipe diameter. Have your code choose the appropriate viscosity value for the calculations based on the user input. Calculate and print the Reynolds number and whether the flow is laminar, in transition, or turbulent. For this problem, it's probably a good idea to print which fluid was used and then echo the input values for velocity and pipe diameter.

*(continued, next page)*

## Program 3

☑ *Prepare and code following the input, calculate, output method, using well-formatted prompts and results.*
☑ *Assign values to variables, and give variables appropriate names for their use.*
☑ *Use boolean and conditional statements.*

A quadratic equation is an equation of the form $Ax^2+Bx+C = 0$. A, B, and C are the coefficients of the equation, and the roots are the values of x at which the equation evaluates to 0. The well-known quadratic formula is often used to find these roots.

Write a program that asks a user for the coefficients A, B and C, and outputs the roots of that equation.

- Use descriptive prompts and output.
- Handle the cases where some or all of the coefficients are 0.
- If the roots have an imaginary component, use i when representing the imaginary term in the output. For example, you may output "3 + 7i" as a root.
- Handle cases in which any or all coefficients are equal to zero
    - If A != 0, there could be 2 real distinct roots, 2 real identical roots (one root of multiplicity 2), or complex roots.
    - If A = 0, we are left with Bx + C = 0, a linear equation with one root.
    - If A = B = 0, we are left with C = 0, so if user entered a non-zero value of C, write a message to the screen indicating this error.

## Program 4

☑ *Create a plan, well-designed test cases, and write a program according to incremental coding practices.*
☑ *Prepare and code following the input, calculate, output method, using well-formatted prompts and results.*
☑ *Assign values to variables, and give variables appropriate names for their use.*
☑ *Use boolean and conditional statements.*

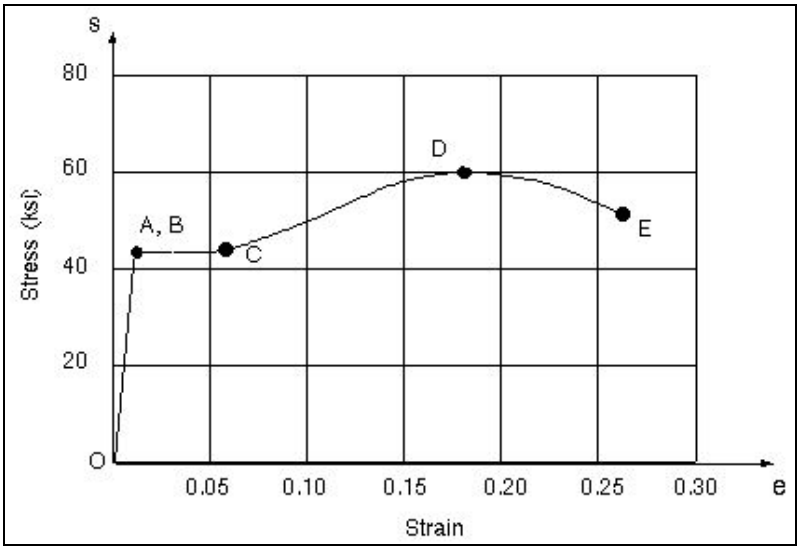
Follow a similar strategy to the one pursued for your team program. *(Note: there will be fewer conditions and fewer necessary test cases. However, there will be more computation, including interpolating data.)*

You will first create a simplified model for the relationship between stress and strain, and then create a program utilizing this model to report the calculated stress for a given strain for structural steel.

*Background*

Stress-strain relationships are important in many engineering disciplines. In rough terms, the strain of an object tells how much it has deformed, and the stress on an object tells how much force the material is exerting in response.

Stress-strain curves display these relationships, plotting strain on the x-axis and stress on the y-axis. For our example, *www.ce.memphis.edu/1101/notes/concrete/section_1_strength_of_materials.html* includes a more complete description. An image web search on "Stress-strain curve for steel" will produce additional examples, both idealized and measured. Different materials will have different stress-strain relationships and curves.




*(continued, next page)*

*Part 1: Preparation*

For this part of the project, create your planning document. You will submit a PDF version with your program.

Examine the stress-strain curve:

- **O – A** An increasing linear elastic region in which the stress is directly proportional to the strain. The slope of this region is called Young's Modulus.
- **A – B** Here there continues to be increasing stress per strain, but not at a constant rate. Points A and B are actually distinct, but in this idealized model, we will ignore this.
- **B – C** From the upper yield point (B) to the lower yield point (C) is a "plastic" region (this is not actually linear, although it appears so in the figure above).
- **C – D** The "strain hardening" region (C to D) applies up to the point of "maximum strength (D)
- **D – E** The "necking" region (D to E), applies up until the fracture point (E).

A. Create a simplified linear model of the stress-strain curve. That is, approximate the curve by a series of at least four (4) linear segments. The lines should begin at [0, 0], and end at the fracture point.
   - For the linear elastic region, calculate the value of Young's Modulus (this should also be a program output).
   - Record the endpoints (as strain and stress "coordinates") for each of the linear segments. You will need to estimate these from the graph.
   - Using more lines will give a more accurate representation, but will be more work in coding; for this assignment, you can use four (4) segments.

B. Next consider what values will be needed for a program that calculates the stress value from a given strain value. Make a list of variables you believe you are likely to need, and the names you will use.

C. Plan the program procedure and create a sequence of steps.
   - When you have a conditional statement, you might want to indicate each part of the condition as a separate action.
   - Calculating the required output will involve linear interpolation (recall Lab 02).
   - The computation will involve multiple stages; separate these (do not just write, "Compute stress").

D. Create a list of test cases for your program, including both "typical" and "edge" cases. Do this before writing the program itself!
   - You should create "intermediate" test cases.
   - Clearly include both "typical" and "edge/corner" cases.
   - You should come up with a complete set of test cases that thoroughly test the idea.
   - For each test case, give a VERY BRIEF statement including:
     - What part of the program it is testing.
     - Is it a "typical" or "edge/corner" case.
     - What is the data you are testing, which region, and what is the expected output value.

*Part 2: Coding*

Only *after* completing Part 1, code your program. Prompt the user for a strain value, and report the stress. As you write your program:

- Include comments for your program. An easy way is to copy your list of steps into PyCharm, and convert them all into comments ["Code" menu → "Comment with Line Comment"].
- Remove and condense comments as you write the code.
- Develop incrementally. That is, write some code, and test it *before* writing the next section.
- Verify that your program runs and passes *all* test cases. You should, in the process of developing, test *every* test case you put together in Part 1.
- Include specific instructions to the user, and write well-formatted descriptive output.