ENGR 102 -Lab #5

Activity #1: Assignment: Writing and testing a larger program

In engineering and science, it is common that we want to calculate the effect of some complex behavior. To make this possible, we create a **model** of the behavior. A model attempts to describe the behavior in a way that is understandable and computable. Some models are based on physical laws and principles, some are based on replicating observations, and many are a combination of these. Once you have a model, you can use it to analyze and predict the performance or behavior of some system or phenomenon.

In this case, you will use a model that has been developed to estimate someone's 10-year risk for likelihood of a heart attack, developed by the NIH. This model was constructed from data analysis of various factors that have been demonstrated to contribute to heart attacks for those ages 20-70.

The process is outlined on the final page of this document:
https://www.nhlbi.nih.gov/sites/default/files/publications/05-3290.pdf
Note that the model works by taking several factors, assigning "point" values to them, and calculating an overall risk based on the sum of those points.

Activity 1: Thinking about the program

Begin by putting together a document that your team will use to analyze the problem. This should be done BEFORE any coding. This document should be saved as a pdf and completed before writing the program Activity 2.

First, your team should review the table on the last page of the document referenced above, and ensure you understand how it works.

Begin by considering 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 believe you are likely to need, and the names you will use to store each.
- Create a sequence of steps that you will follow
    - Each step should be a short description of the goal of the step. The steps should not be code, or simple versions of code, but rather a description of the purpose of an action. Each step should be the equivalent of a few lines of code.
        - E.g. "Compute points based on age" might be a good description of a step.
    - If you have a conditional statement (and you should have several…), you might want to indicate each part of the condition as a separate action.

Next, determine a set of test cases.

- 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!
- You can create "intermediate" test cases. That is, your test cases do not have to be for the entire program, but you could write a test to verify that a particular part of your program is working.

- For this program, a "good" set of test cases would involve well over 100 tests, since many of the values to test are just to be pulled from tables. Since this is rather cumbersome, you do not have to give a full set, but
    - Your team should have at least 40 different test cases (more is fine)
    - Your test cases should clearly include both "typical" and "edge/corner" cases.
    - Your test cases should address each part of the program (e.g. don't just test the age-based points for men).
- For each test case, you should give a VERY BRIEF statement of what the test is covering (i.e. what part of the program is it testing, and is it a "typical" or "edge/corner" case, what the data is you are testing on, and then what result you expect.
    - E.g. "Age-based score (typical); Sex: Male, Age: 62; Age-based score: 10"
    - E.g. "Entire program (extreme low); Sex: Female, Age: 20, etc. etc.; 10-year risk: <10%)"
- You will probably find it easiest to divide up the work among your team for creating test cases, and then putting them all into a table.

When you have done this, save the document in PDF format.

Activity 2: Constructing your program

As a team, construct your program. As you do this, please be sure to do the following:

- Include comments for your program. You should probably begin by converting your list of steps into comments.
- Develop incrementally. That is, write some code, and test it before writing the next section of code. Don't let your teammates skip this!
- Be sure your program runs and passes all test cases. You should, in the process of developing, test every single one of the test cases you put together in Activity 1.
- Be sure to include specific instructions to the user for getting input, and write a descriptive output.