CSE 101: Computer Science Principles (Fall 2019)

Lab #1

Assignment Due: Saturday, September 21, 2019, by 11:59 PM

Assignment Objectives

By the end of this assignment you should be able to develop original Python functions to solve simple programming problems involving variables, arithmetic, and conditional statements.

Getting Started

This assignment requires you to write Python code to solve several computational problems. To help you get started, we will give you a basic starter file for each problem. These files will contain *function stubs*[1] and a few tests you can try out to see if your code seems to be correct (**note that the test cases we give you in the starter files are just examples; we will use different test inputs to grade your work!**). You need to complete (fill in the bodies of) these functions for the assignments. **Do not, under any circumstances, change the names of the functions or their parameter lists.** The automated grading system will be looking for functions with those exact names and parameter lists; if you change any function headers, the grading program will reject your solution and mark it as incorrect.

Directions

Solve each of the following problems to the best of your ability. We have provided a code skeleton for each of the programming problems. The automated grading system will execute your solution to each programming problem several times, using different input values each time. Each test that produces the correct/expected output will earn 1 or more points. This assignment contains 2 problems, and is worth a total of 20 points. **Note that not every problem may be worth the same number of points!**

- ⚠ Each starter file has a comment block at the top with various important information. Be sure to add your information (name, ID number, and NetID) to the first three comment lines, so that we can easily identify your work. **If you leave this information out, you may not receive credit for your work!**
- ⚠ Submit your work as a set of individual files (one per problem). **DO NOT** zip or otherwise compress your files, and **DO NOT** place all of your work in a single file. If you do so, we may not be able to grade your work and you will receive a failing grade for this assignment!
- ⚠ Every function **MUST** use the names and parameter lists indicated in the starter code file. Submissions that have the wrong function names (or whose functions contain the wrong number of parameters) can't be graded by our automated grading system, and may receive a grading penalty (or may not be graded at all).
- ⚠ Every function must explicitly ***return*** its final answer; the grading program will ignore anything that your code prints out. Along those lines, do **NOT** use `input()` anywhere within your functions (or anywhere before the `if __name__ == "__main__":` statement); your functions should get all of their input from their parameters. Programs that crash will likely receive a failing grade, so test your code thoroughly **with Python 3.7.4 or later** before submitting it.
- ⚠ Blackboard will provide information on how to submit this assignment. You **MUST** submit your completed work as directed by the indicated due date and time. We will not accept (or grade) any work that is submitted after the due date and time, or that is submitted before you have signed the course's Academic Honesty agreement.
- ⚠ **ALL** of the material that you submit (for each problem) **MUST** be your own work! You may not receive assistance from or share any materials with anyone else, except for the instructor and the (current) course TAs.

[1] Stubs are functions that have no bodies, or have very minimal, bodies

Part I: Retirement Calculations (10 points)

(Place your answer to this problem in the "retirement.py" file)

Complete the `retirement()` function, which takes three integer arguments: your current age, the age at which you'd like to retire, and the current year. The function calculates how many years are left before you can retire, and returns an integer representing the year in which you can retire. For example, `retirement(25, 65, 2019)` would return 2059 (if you're 25 now and want to retire at age 65, you have 40 years between now and retirement; 2019 plus 40 is 2059).

You may assume that all three arguments are positive, and that the retirement age is strictly greater than your current age.

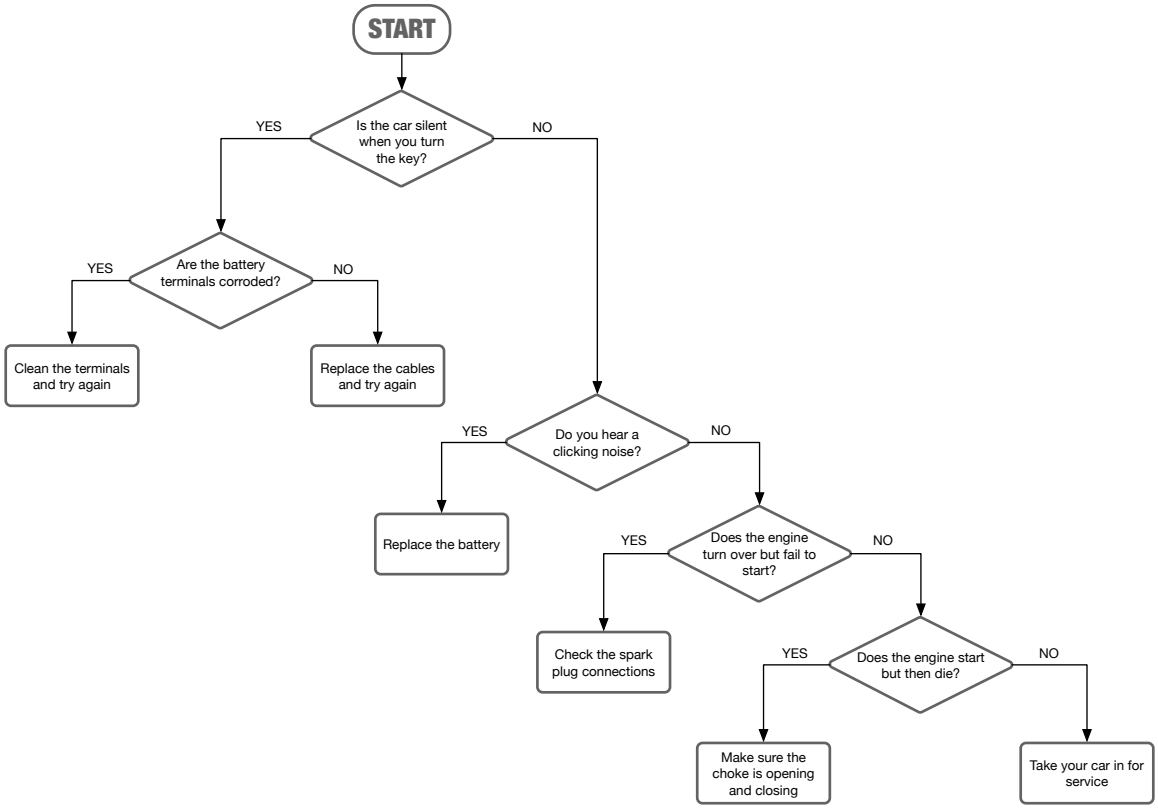
Examples:

Function Call	Return Value
`retirement(25, 65, 2019)`	`2059`
`retirement(55, 62, 2022)`	`2029`
`retirement(30, 78, 2014)`	`2062`

Part II: Car Trouble (10 points)

(Place your answer to this problem in the "car_trouble.py" file)

Complete the `car_trouble()` function, which takes four single-character strings as arguments (each string is either `"Y"` or `"N"`, and is always uppercase). The function uses a set of conditional (`if` and `else`) statements to process the arguments and return a string based on the flowchart below:



You may assume that the function will always be called with a legal set of arguments (in other words, it will always be called with a combination of values that lead to one of the six troubleshooting messages from the flowchart). Use the following text for each of the possible return values (you must match the spelling and capitalization **EXACTLY**; note that there is exactly one space between each word):

- `"clean terminals"`
- `"replace cables"`
- `"replace battery"`
- `"check spark plugs"`
- `"ensure choke is opening and closing"`
- `"bring it in for service"`

For example, `car_trouble("N", "N", "Y", "N")` would return the (all lowercase) string `"check spark plugs"` (in this case, the fourth argument is ignored).

Examples:

Function Call	**Return Value (a string)**
`car_trouble("Y", "Y", "N", "Y")`	`clean terminals`
`car_trouble("Y", "N", "Y", "Y")`	`replace cables`
`car_trouble("N", "Y", "N", "N")`	`replace battery`
`car_trouble("N", "N", "N", "N")`	`bring it in for service`
`car_trouble("N", "N", "N", "Y")`	`ensure choke is opening and closing`