Lab 03 Specification – Functions and Recursion
Due (via your git repo) no later than 8 a.m., Friday, 21 September 2018.
50 points

Lab Goals

- Practice usage of Function in C
- Write a recursive C program

Suggestions for Success

- Take a look at the suggestions for successfully completing the lab assignment, which is available at: `https://www.cs.allegheny.edu/sites/amohan/resources/suggestions.pdf`

Learning Assignment

To do well on this assignment, you should also read

- KR chapter 01 - Section 1.7
- KR chapter 04 - Section [4.1 - 4.3] [4.5] [4.9 - 4.10]
- Class discussion notes, slides, and in-class coding files.

Assignment Details

In this lab assignment, you will work on modularizing programs using Functions in C. Next you will get a chance to work on recursive C programs.

Part 1: Quiz Simulator (20 points)

Write a program called QuizSim.c, that simulate **CS200-Quiz-01** given earlier this week. A student using your quiz simulator should be able to take the quiz and his score should be provided at the end of the quiz. In addition to the quiz score, a brief feedback should also be presented to the student at the end of the quiz.

Okay, let us look at some more details regarding the quiz simulator program.

1. Your program should have a main function, that should print the following message as a starting point:

   ”Welcome to the CS200 Fall 2018 Quiz 01! Let’s begin.”

2. There should be a total of 12 functions in your program, including the main function. Here we are assuming there are 10 questions and we have a function for each corresponding question [As the case in quiz #1]. There is a function called ”outputNumCorrect”, that will output the score and the feedback on the quiz. The function prototypes for the 11 functions placed in the main() funciton are provided below:

   ```
   // function prototypes
   int getQuestion1(void);
   int getQuestion2(void);
   int getQuestion3(void);
   int getQuestion4(void);
   int getQuestion5(void);
   int getQuestion6(void);
   int getQuestion7(void);
   int getQuestion8(void);
   int getQuestion9(void);
   int getQuestion10(void);
   void outputNumCorrect(int);
   ```

3. The series of functions named ”getQuestion”, should display the question and the possible answers. Each individual answer should have an integer value associated with it. Then, prompt the student to choose a correct answer. A correct answer is chosen, by providing an integer value that matches the answer. The output of these functions should be an integer value that matches the correct answer. Assume all the questions in the quiz are multiple choice questions.

   For example, I would display the question and the possible answers similar to the text shown below:

   Which of these should you NOT do to help stop global warming?

   1: Buy more frozen foods

   2: Fly less

   3: Use a clothesline instead of a dryer

   4: Cover pots while cooking

   You should note that this is just an example, probably a quiz question about Global Weather. In your program, the questions and the possible answers should all be coming directly from the Quiz #1.

4. In the main function, every time a function named ”getQuestion” is called, a counter should be maintained, that keeps track of the correct answers based on the value returned by the 10 functions named ”getQuestion”.

5. The function called ”outputNumCorrect” should display the total number of correct answers to the student.

   If the student answered all 10 questions correctly, then congratulate the student with a message ”Excellent, you have answered all the questions correctly. You received 10 out of 10!”

   If the student answered [8 or 9] questions correctly, then greet the student with a message ”Very good. You received X out of 10!” Here X should be replaced with the students quiz score.

   If the student answered [6 or 7] questions correctly, then greet the student with a message ”Good. You received X out of 10!” Here X should be replaced with the students quiz score.

   If the student answered [<= 5] questions correctly, then display the following message to the student: ”Time to brush up on your knowledge of the topics covered in the Quiz!” Additionally, point out the link to the slides that covered the topics in the quiz. The link (URL) should be added to the display message above.

Part 2: Recursive Palindrome Checker (15 points)

Write a program called RecursivePalindrome.c, that is a variation of your Lab 02 Palindrome program. You should use recursion in your program called Palindrome.c that performs the following functionality:

1. prompt the user to type in an input string and store the string into a variable firstString.
2. reverse the string *firstString* and store it into a new variable called *secondString*.
3. compare both strings firstString and secondString and validate if they are equal. If they are equal then print ”Strings are palindrome”, else print ”Strings are not palindrome”. For example ”DAD”, ”MOM”, ”CIVIC”, ”KAYAK”, etc.. are palindromes.
4. You are only allowed to use recursion to implement this part. Any solution that uses iterative approach would not be acceptable. Additionally, the usage of any external string libraries, implementation using strcmp, pointers, and other library based functions is also not acceptable.
5. As we did in the class discussion, draw the recursive calls hierarchy and a tree to show all the return values from those individual calls. This visualization process will guide you to think effectively on how to implement this program. It is **not required** to show the your visualization work in the lab submission. Further, if you practice the visualization steps, it will also help you to do better in the quiz and exams.
6. Clearly state the base case of your recursion by adding appropriate comments in your code.

Part 3: Sum of digits calculator (15 points)

Write a program called SOD.c, that computes the **S**um **O**f **D**igits in a user provided input. You should only use recursion to implement your program. Any iterative solution is not acceptable. Here are some more details about the program requirement:

1. You can assume the user input is going to be a 5-digit positive integer.
2. As we did in the class discussion, draw the recursive calls hierarchy and a tree to show all the return values from those individual calls. This visualization process will guide you to think effectively on how to implement this program. It is **not required** to show your visualization work in the lab submission. Additionally, if you practice the visualization steps, it will also help you to do better in the quiz and exams.
3. Clearly state the base case of your recursion by adding appropriate comments in your code.

Submission Details

1. Your lab assignment solution file must be uploaded to your repository by the due date and time.
2. Questions about the lab? Bring them to class on Tuesday morning!

Before you turn in this assignment, you must ensure that the course instructor has read access to your git repository that is named according to the convention `cs200F2018-<your user name>`.

Grading Rubric

1. If you complete Part 1 fully, as per the requirement outlined above, you will receive a total of 20 points.
2. If you complete Part 2 fully, as per the requirement outlined above, you will receive a total of 15 points.
3. If you complete Part 3 fully, as per the requirement outlined above, you will receive a total of 15 points.
4. Failure to upload the lab assignment code to your git repo, will lead you to receive "0" points given for the lab submission.
5. There will be a partial credit (30% of the points allocated for the part) given if your code in (Part 1, 2 and 3) compiles correctly but not able to produce the expected result.
6. There will be no partial credit awarded if your code doesn't compile correctly. It is highly recommended to validate if the correct version of the code is being submitted before due date and make sure to follow the honor code policy described in the syllabus.
7. If you needed any clarification on your lab grade, talk to the instructor. The lab grade may be changed if deemed appropriate.