CS 210 - Fundamentals of Programming I
Fall 2019 - Programming Project 3
20 points

Out: September 12, 2019
Due: September 19, 2015 (Tuesday)

Reminder: This is a programming project, and work on this project should be done individually. Assistance from other students is limited to questions about specific issues as noted in the syllabus.

Problem Statement
A common programming assignment, usually called the Guessing Game, is to have the computer pick a random number and then the user guesses until they find it, as was demonstrated in lecture. In the Reverse Guessing Game, this is turned around. The human user decides on a number between 1 and 1000 and then computer tries to guess the number repeatedly with the user telling the computer whether its guess is too high, too low, or correct.

Program Specifications
Like the last assignment, you are given the program specification in the form of an analysis and design for a series of functions that must be implemented. Each function accomplishes one task needed for this program. You should write a function and then test it with a `main` program that ensures that the function works by itself. Once the function is working, move on to the next function, which will require a different `main` function to test it. Once all of the individual functions work, the final `main` function that actually will play the game is written.

Function: print_greeting
Analysis:

Objects	Type	Movement	Name
minimum possible guess value	int	received	min
maximum possible guess value	int	received	max
maximum number of computer guesses	int	received	guesses

Design: This function's task is simply to print out the greeting at the beginning of the program stating the range of the numbers that may be chosen by the user and the maximum number of guesses the computer is allowed to make. See the sample run for the exact text format.

Function: user_wishes_to_continue
Analysis, Design, and Implementation covered in lecture on September 12. See the sample run for the exact text format of the prompt and error messages.

Function: prompt_user

Analysis:

Objects	Type	Movement	Name
computer's guess	int	received	guess
the user's response to the guess	char	returned	response

Design: This function's task is to tell the user the computer's guess and ask the user whether it is too high (H), too low (L), or the correct value (E for exact). The function should allow the user to entered an H, an L or an E in upper or lowercase and return the character entered in uppercase. It should also check for invalid input and ask the user again repeatedly until valid input is entered. This loop is similar to the loop in the user_wishes_to_continue function demonstrated in lecture. See the sample run for the exact text format of the prompt and error message.

Function: print_results

Analysis:

Objects	Type	Movement	Name
number of computer wins	int	received	comp_wins
number of user wins	int	received	user_wins

Design: This function's task is to print out the number of wins by the computer and by the user, and an ending message. See the sample run for the exact text format.

Function: play_game

Analysis:

Objects	Type	Movement	Name
minimum possible guess value	int	received	min
maximum possible guess value	int	received	max
maximum number of computer guesses	int	received	guesses
result of game	int (boolean)	returned	has_won

Design: This function's task is to play one game of the reverse guessing game. The basic idea is that the computer keeps track of the possible lowest and highest values it knows the user could have picked. Each guess is the number in the middle of the range. If the guess is too high or too low, the range is adjusted as shown below. The computer stops guessing when the maximum number of guesses has been reached or the user says it has guessed the number. The function returns true (1) if the computer guessed the number and false (0) if it runs out of guesses. Here are the steps:

1. Initialize local variables low and high to min and max, respectively
2. Initialize local variable num_guesses to 0
3. Initialize the boolean result variable has_won to 0 (i.e. false, to indicate the game has not, yet, been won by the computer)
4. While the computer still has guesses left and it has not won yet
    a) Compute a guess as the average of high and low
    b) Get a response from the user using function prompt_user
    c) If the user responds with H (the guess is too high), the new value of high becomes guess-1

    d) If the user responds with L (the guess is too low), the new value of low becomes guess+1
    e) If the user responds with E (the guess is exact), the has_won flag is set to 1 (i.e., true, to indicate that the computer has won)
    f) Increment num_guesses
5. If the has_won flag is true, print "I win!" and the number of guesses, otherwise print "You win!
6. Return has_won flag

Main Program

Analysis:

Objects	Type	Name
number of wins by the computer	int	comp_wins
number of wins by the user	int	user_wins

Design: Finally we arrive at the main function of the program, now that we have all the pieces. For this assignment, the minimum and maximum values that can be chosen by the user is 1 to 1000, inclusive, and the maximum number of guesses that computer can make is 5. (Something to think about: how many guesses would the computer need to find the user's number every time?)

1. Print the greeting using the print_greeting function
2. Initialize the counter variables to 0
3. In a loop, play a game using the play_game function, increment the win count of the winner of the game, and increment the number of games played until the user does not wish to continue. Use the user_wishes_to_continue function to control the loop.
4. Print the results and ending message using the print_results function
5. Return 0 to exit the program

Assignment

Write a C program that implements the Reverse Guessing Game. **Your program must follow the specifications given above to earn full credit.** That is, your program must define and use at least the specified functions. (The names of functions and variables do not have to be exactly the same.)

The output of the program must conform exactly to the following example runs (there are 4 separate runs shown; user input shown in **bold**). Note there is blank line before the "Do you wish to continue..." prompt, and there is one space after the first sentence punctuation in a two sentence line of output. (E.g. in the error messages.) And as usual, there must be a newline at the end of the output.

```
Welcome to the (reverse) guessing game.
Choose a number between 1 and 1000 inclusive
and I will try to guess it in 5 or fewer tries

My guess is 500.
Is my guess too (H)igh, too (L)ow, or (E)qual to your number? E
I win! It took me 1 guess(es).

Do you wish to continue (y/n)? y
My guess is 500.
Is my guess too (H)igh, too (L)ow, or (E)qual to your number? h
My guess is 250.
```

```
Is my guess too (H)igh, too (L)ow, or (E)qual to your number? H
My guess is 125.
Is my guess too (H)igh, too (L)ow, or (E)qual to your number? l
My guess is 187.
Is my guess too (H)igh, too (L)ow, or (E)qual to your number? L
My guess is 218.
Is my guess too (H)igh, too (L)ow, or (E)qual to your number? h
You win!

Do you wish to continue (y/n)? y
My guess is 500.
Is my guess too (H)igh, too (L)ow, or (E)qual to your number? l
My guess is 750.
Is my guess too (H)igh, too (L)ow, or (E)qual to your number? l
My guess is 875.
Is my guess too (H)igh, too (L)ow, or (E)qual to your number? Q
Bad Input. Try again.
Is my guess too (H)igh, too (L)ow, or (E)qual to your number? l
My guess is 938.
Is my guess too (H)igh, too (L)ow, or (E)qual to your number? e
I win! It took me 4 guess(es).

Do you wish to continue (y/n)? p
Bad Input. Try again.

Do you wish to continue (y/n)? n
I won 2 game(s).
You won 1 game(s).
Thank you for playing!
```

REMINDER: Your program must compile for it to be graded. Submissions that do not compile will be returned for resubmission and assessed a late penalty. Submissions that do not substantially work also will be returned for resubmission and assessed a late penalty.

Follow the program documentation guidelines in the *C Programming Style Guideline* handout. As stated in the syllabus, part of the grade on a programming assignment depends on how well you adhere to the guidelines. The grader will look at your code and grade it according to the guidelines. **Be sure to run the Source code formatter (Astyle) plugin before you submit your program.**

What to Submit

Electronically submit a zipfile containing **`main.c`** (only) as explained in class and in the handout *Submission Instructions for CS 210*. The submission system will start accepting assignments no earlier than the evening of Saturday, September 14. Reminders: you may submit as many times as needed, and only the last submission will be graded. All assignments are due by 11:59pm to earn full credit.