COP 3223 Program #5: Factoring Games (Functions)

Due Date: *Please Consult WebCourses*

Objective

1. To give students practice in writing functions.

Problem: Factoring Game(s)

You've been having some trouble with arithmetic on your no calculator exams and have decided that you're going to write a little program to help you study factoring integers.

The game will give the user two possible games to play:

1) A game where the user has to come up with any non-trivial factorization of an integer.

2) A game where the user has to determine the number of non-trivial factorizations of an integer.

The definition of a non-trivial factorization, of a positive integer n, for the purposes of this homework assignment is a pair of integers, a and b, such that n = a x b and a>1 and b≥a.

For example, 8 has only one non-trivial factorization, 2 x 4, and 36 has 4 non-trivial factorizations: 2 x 18, 3 x 12, 4 x 9 and 6 x 6.

Thus, the menu you'll present the user looks like this:

```
Please select one of the following menu choices.\n");
1. Factorize a random integer.
2. Determine the number of non-trivial factorizations of an integer.
3. Quit.
```

If the user enters an invalid choice, give them the following prompt:

```
Sorry, that is not a valid choice.
```

and then reprompt them with the menu again. Continue doing so until they enter 1, 2 or 3.

If the user enters 1 or 2, your program should then prompt him for the maximum value of the number to be factorized. **You may assume that the user enters a valid integer 4 or greater.**

For each game, you'll generate a random integer in between 4 and the maximum limit given **that is not a prime number.**

From there, you'll prompt the user to either factorize that randomly chosen integer, or determine the number of non-trivial factorizations it has, depending on their menu choice.

After reading in the user's answer (2 integers for the first type of problem, 1 integer for the second type of problem), determine if it's correct or not and print out a message accordingly. (Specific formats are provided with code scaffold given.)

Keep track of how many questions the user tries to answer before quitting as well as how many they get correct. When they quit print out their number correct, the total number they tried and the percentage of problems they solved correctly.

Required Functions
The main function has already been written for you in factorgame_scaffold.c.

All you have to do for this assignment is fill in the following functions. The signatures as well as the corresponding pre-conditions and post-conditions are provided below:

```
// Pre-condition: n is a positive integer.
// Post-condition: Returns 1 if n is a prime number, 0
// otherwise.
int isPrime(int n);

// Pre-condition: limit is an integer greater than or equal
// to 4, and less than 32767.
// Post-condition: Returns a randomly chosen composite (not
// prime) integer in between 4 and limit, inclusive.
int randNonPrime(int limit);

// Pre-condition: n is a positive integer, a and b are
// integers
// Post-condition: Returns 1 if a and b are both positive
// and n = a x b and a <= b, returns 0 otherwise.
int isValidFact(int n, int a, int b);

// Pre-condition: n is a positive integer in between 4 and
// 32767.
// Post-condition: Returns the number of unique integers a,
// greater than 1 for which n = a x b and a <= b.
int numValidFact(int n);

// Pre-condition: none
// Post-condition: Prompts the user with a menu of choices,
// and continues to do so until the user chooses a valid
// choice: 1, 2 or 3. Then, that valid choice is returned.
int menu();
```

Sample Input/Output
Will be provided in a separately posted file.

Grading Details

Unlike other assignments where there are lots of execution points, this assignment will be graded upon following the details specified in the post-conditions of each function filled out. The reason for this is that by and large, the program can appear to be running correctly even when the functions are not written to specification. Thus, the graders will have to judge correctness (and your grade) analytically (by looking at your code) and you'll have to do the same before submitting.

As an example, it's possible to run several games and get always get a composite number for the game, but for your function that generates a random composite number not to be correct (and for it to occasionally incorrectly generate a prime). Most real life code is like this, where incorrect code can hide and run correctly for some time.

Deliverables

Please submit a single .c file called, factorgame.c. Please leave the internal comments in main, but add your own internal comments inside of the functions. Please edit the header comment to include your name and what you added (function definitions).