CECS 130-01&02
Spring 2017
Assignment 4
Assignment due on or before 9:00 pm Friday, February 17, 2017

1. Do challenge 2 of chapter 4 on page 106 (See image at end of this document).
2. Do challenge 3 of chapter 4 on page 106.
3. Do challenge 1 of chapter 5 on page 126.
4. Do challenge 2 of chapter 5 on page 126 and add a main() function that calls each of the three functions you defined.

Please see figures at end of assignment

A note about assignments and reports:

Your presentation in your reports and assignments reflects great deal about you, your understanding of the assignment and on how much this course means to you. I try very hard to look at the substance of the report but I will be lying if I said that presentation does not influence my judgment. It would be wise on your part to assume that this true in every course at school and in real life/work. I expect your reports to be well formed and conform to the following rules:

1. All reports have to be submitted as a **PDF** report that contains:
   1.1. Title page with your name, assignment number and the day you are actually submitting this report (Not the assignment due date)
   1.2. A comprehensive set of snapshots showing the inputs submitted and outputs obtained in the case of a successful output or a failure.
2. A text file that contains all source code, please concatenate all source code in one text file.
3. Make sure that you include as a comment at the top of your file your name and section:
   As an example:

```
/*************************/
/* John Q. Public        */
/* CECS 130-11           */
/* Assignment 35         */
/*************************/
```

   Failure to do this will cost you points.

4. Please zip both the PDF document with the source code and submit one zip file.
5. Please do not submit your eclipse or bloodshed project or any IDE project that you may be using. I will be compiling and testing your source code from the text file in part 2 above to test running your applications and to verify that they run.
6. Remember that you must only access BlackBoard using section 130-01

Challenges

1. Create a counting program that counts from 1 to 100 in increments of 5.
2. Create a counting program that counts backward from 100 to 1 in increments of 10.
3. Create a counting program that prompts the user for three inputs (shown next) that determine how and what to count. Store the user's answers in variables. Use the acquired data to build your counting program with a `for` loop and display the results to the user:
   - Beginning number to start counting from
   - Ending number to stop counting at
   - Increment number
4. Create a math quiz program that prompts the user for how many questions to ask. The program should congratulate the player if he gets the correct answer or alert the user of the correct answer if he answers the question incorrectly. The math quiz program should also keep track of how many questions the player has answered correctly and incorrectly and display these running totals at the end of the quiz.
5. Modify the Concentration game to use a main menu. The menu should allow the user to select a level of difficulty or quit the game. (A sample menu is shown next.) The level of difficulty could be determined by how many separate numbers the user has to concentrate on or how many seconds the player has to memorize the sequence of numbers. Each time the user completes a single game of Concentration, the menu should reappear, allowing the user to continue at the same level, continue at a new level, or simply quit the game.
   1. Easy (remember 3 numbers displayed for 5 seconds)
   2. Intermediate (remember 5 numbers displayed for 5 seconds)
   3. Difficult (remember 5 numbers displayed for 2 seconds)
   4. Quit

Chapter 4 challenges (Page 106)

Challenges

1. Write a function prototype for the following components:
   - A function that divides two numbers and returns the remainder
   - A function that finds the larger of two numbers and returns the result
   - A function that prints an ATM menu—it receives no parameters and returns no value
2. Build the function definitions for each preceding function prototype.
3. Add your own trivia categories to the Trivia game.
4. Modify the Trivia game to track the number of times a user gets an answer correct and incorrect. When the user quits the program, display the number of correct and incorrect answers. Consider using global variables to track the number of questions answered, the number answered correctly, and the number answered incorrectly.

Chapter 5 challenges (Page 126)