CECS 130
Assignment 3
Assignment due on or before 9:00pm Wednesday, June 7, 2017

1. Do challenge 1 of chapter 6 on page 146.
2. Do challenge 2 of chapter 6 on page 146.
3. Do challenge 1 of chapter 7 on page 169.
4. Do challenge 3 of chapter 7 on page 169.
5. Do challenge 1 of Chapter 8 on page 192.
6. Do challenge 3 of Chapter 8 on page 192.

See images at end of assignment sheet

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. Build a program that uses a single-dimension array to store 10 numbers input by a user. After inputting the numbers, the user should see a menu with two options to sort and print the 10 numbers in ascending or descending order.
2. Create a student GPA average calculator. The program should prompt the user to enter up to 30 GPAs, which are stored in a single-dimension array. Each time the user enters a GPA, he should have the option to calculate the current GPA average or enter another GPA. Sample data for this program is shown here:

   GPA: 3.5

   GPA: 2.8

   GPA: 3.0

   GPA: 2.5

   GPA: 4.0

   GPA: 3.7

   GPA Average: 3.25

   Hint: Be careful not to calculate empty array elements into your student GPA average.
3. Create a program that allows a user to enter up to five names of friends. Use a two-dimensional array to store the friends' names. After each name is entered, the user should have the option to enter another name or print out a report that shows each name entered thus far.
4. Modify the Tic-Tac-Toe game to use a two-dimensional array instead of a single-dimension array.
5. Modify the Tic-Tac-Toe program or build your own Tic-Tac-Toe game to be a single-player game. (The user will play against the computer.)

Challenges

1. Build a program that performs the following operations:
   - Declares three pointer variables called `iPtr` of type `int`, `cPtr` of type `char`, and `fFloat` of type `float`
   - Declares three new variables called `iNumber` of `int` type, `fNumber` of `float` type, and `cCharacter` of `char` type
   - Assigns the address of each nonpointer variable to the matching pointer variable
   - Prints the value of each nonpointer variable
   - Prints the value of each pointer variable
   - Prints the address of each nonpointer variable
   - Prints the address of each pointer variable

2. Create a program that allows a user to select one of the following four menu options:
   - Enter New Integer Value
   - Print Pointer Address
   - Print Integer Address
   - Print Integer Value

   For this program, you need to create two variables: one integer data type and one pointer. Using indirection, assign any new integer value that the user enters through an appropriate pointer.

3. Create a dice rolling game. The game should allow a user to toss up to six dice at a time. Each toss of a die will be stored in a six-element integer array. The array is created in the `main()` function but passed to a new function called `TossDie()`. The `TossDie()` function will take care of generating random numbers from one to six and assigning them to the appropriate array element number.

4. Modify the Cryptogram program to use a different type of key system or algorithm. Consider using a user-defined key or a different character set.

Challenges

1. Create a program that performs the following functions:
   - Uses character arrays to read a user's name from standard input
   - Tells the user how many characters are in her name
   - Displays the user's name in uppercase
2. Create a program that uses the `strstr()` function to search the string, "When the going gets tough, the tough stay put!" for the following occurrences (display each occurrence found to standard output):
   - `"Going"`
   - `"tou"`
   - `"ay put!"`
3. Build a program that uses an array of strings to store the following names:
   - `"Florida"`
   - `"Oregon"`
   - `"California"`
   - `"Georgia"`

   Using the preceding array of strings, write your own `sort()` function to display each state's name in alphabetical order using the `strcmp()` function.
4. Modify the Word Find game to include one or more of the following suggestions:
   - Add a menu to the Word Find game that allows the user to select a level of difficulty, such as beginning, intermediate, and advanced. The number of seconds the user has to guess or the length of the text in which the user will look for words could determine the level of difficulty.
   - Incorporate multiple words into the text areas.
   - Track the player's score. For example, add 1 point for each word guessed correctly, and subtract 1 point for each word guessed incorrectly.
   - Use the `strlen()` function to ensure the user's input string is the same length as the hidden word.