Project[6]: Functions

Due Tuesday, 10/31/2017, 11:59am

Project Goals

The goals of this project are to:

1. Get you familiar with the use of functions.
2. Show you how simple it can be to implement complicated-looking functions.

Important Notes:

1. **Formatting:** Make sure that you follow the precise recommendations for the output content and formatting. Your assignment will be auto-graded and any change in formatting will result in a loss in the grade.
2. **Comments:** Header comments are required on all files and recommended for the rest of the program. Points will be deducted if no header comments are included.

Problem 1

Write a program that simulates a simplified version of the game scrabble. The program should run through the following steps:

1. Generate a set of 7 letters for the user
2. Ask the user to enter his/her word
3. Read the user's word
4. Check the validity of the user's word, to make sure that the user has only used letters that are available.
5. Display the value of the word.
6. Terminate the program.

Constraints: your program should define and use at least the following four functions:

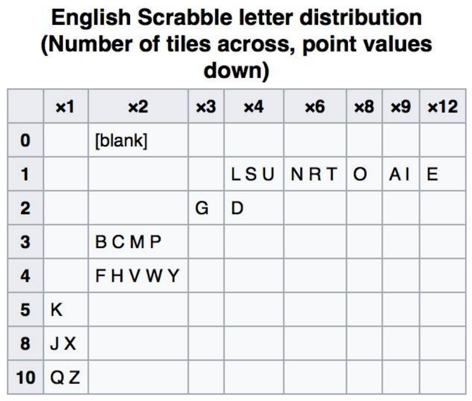
```
void generate_letter_set (int letter_set[], int size_letter_set,
int num_letters)

int read_word (char word[], int max_size_word)

_Bool check_word (char word[], int size_word, int letter_set[],
int size_letter_set)

int compute_word_value (char word[], int size_word)
```

1. **`void generate_letter_set (int letter_set[], int size_letter_set, int num_letters)`**
   This function should use the approach we discussed in our class example to randomly generate a set of letters, based on the number of tiles available (as shown in the table on the side). As in regular scrabble, multiple letters of the same type can be selected, according with the total number available. For instance, since there are only 2 tiles with letter B, the function could return at most two letter Bs. The array `letter_set` should store the counts of how many of either A, B, C, … etc. are "drawn" by the random generator. Parameter `size_letter_set` is the size of the array `letter_set` (this should be the total number of characters in the alphabet). The function should use the information regarding the available number of tiles for each letter (similar to the `const` arrays we used in the random cards example). The `num_letters` parameter is for how many letters need to be drawn. Use 7 as parameter for the value of `num_letters` argument when calling the function (you should use a macro definition for that value).



English Scrabble letter distribution (Number of tiles across, point values down)

	×1	×2	×3	×4	×6	×8	×9	×12
0		[blank]						
1				L S U	N R T	O	A I	E
2			G	D				
3		B C M P						
4		F H V W Y						
5	K							
8	J X							
10	Q Z							

2. **`int read_word (char word[], int max_size_word)`**
   The function should prompt the user to enter the word, then read it (one character at a time) and store it in the array `word`. The function should also return the size of the word entered by the user (i.e., how many letters there were in the word). The value of the `max_size_word` parameter should be 7, as it is usually the case in scrabble.
3. **`_Bool check_word (char word[], int size_word, int letter_set[], int size_letter_set)`**
   The parameter `word` stores the word entered by the user, `size_word` is the size of the word entered, `letter_set` is the array of counts that has been generated by function `generate_letter_set` and `size_letter_set` is the size of the array `letter_set` (this should be the total number of characters in the alphabet). The function should check if the word entered by the user uses proper letters from the generated letter set and should return true or false based on this evaluation.
4. **`int compute_word_value (char word[], int size_word)`**
   The function should use the information from the point values for each of the letter tiles to compute the value of the word. The function should use information regarding the point values for each letter (similar to the `const` arrays we used in the random cards example).

Your program should run as follows (items underlined are entered by the user):

```
This program plays the game of scrabble.
Your letters are: A B F G I N O
Enter your word: BIG
The value of your word is: 6
Thank you for playing.
```

Below is an example of what your program should print if the user enters an illegal word:

```
This program plays the game of scrabble.
Your letters are: A G N Q R S U
Enter your word: SUNS
The word is not valid. Use your letters: A G N Q R S U
Enter your word: SUN
The value of your word is: 3
Thank you for playing.
```

Save your program in a file called `scrabble.c`.

Challenge 1 (10 extra credit points): Modify the above program to run in a loop as long as the user wants to keep playing, running through the following steps:

1. Generate a set of 7 letters for the user
2. Ask the user to enter his/her word
3. Read the user's word
4. Check the validity of the user's word, to make sure that the user has only used letters that are available.
5. Display the value of the word.
6. Ask the user if he/she wants to continue playing. If yes, the program should repeat steps 1 through 7, otherwise it should terminate.

Your program should run as follows (items underlined are entered by the user):

```
This program plays the game of scrabble.
Your letters are: A B F G I N O
Enter your word: BIG
The value of your word is: 6
Do you want to play again (Y/N): Y
Your letters are: A G N Q R S U
Enter your word: SUNS
The word is not valid. Use your letters: A G N Q R S U
Enter your word: SUN
The value of your word is: 3
Do you want to play again (Y/N): N Thank
you for playing.
```

Save your program in a file called `scrabble_c.c`.

Grading Rubric

Grading will be done for each problem as follows:

Correctly-named file	5%
Header comment	2%
Program compiles	5%
Correctly-reading data from terminal	28%
Correct use of functions	30%
Correct result printed	30%

Submission details

To submit your project, you will have to save your project files to nomachine:

- create a directory called “project6”
- save your *.c files in that directory
- TO Submit:

```
> cd project6
> submit
```

The submission script copies all files in the current directory to our directory. You may submit as many times as you like before the deadline, we only keep the last submission.

Academic Honesty

Academic dishonesty is against university as well as the system community standards. Academic dishonesty includes, but is not limited to, the following:

Plagiarism: defined as submitting the language, ideas, thoughts or work of another as one's own; or assisting in the act of plagiarism by allowing one's work to be used in this fashion.

Cheating: defined as (1) obtaining or providing unauthorized information during an examination through verbal, visual or unauthorized use of books, notes, text and other materials; (2) obtaining or providing information concerning all or part of an examination prior to that examination; (3) taking an examination for another student, or arranging for another person to take an exam in one's place; (4) altering or changing test answers after submittal for grading, grades after grades have been awarded, or other academic records once these are official.

Cheating, plagiarism or otherwise obtaining grades under false pretenses” constitute academic dishonesty according to the code of this university. Academic dishonesty will not be tolerated and penalties can include canceling a student’s enrollment without a grade, giving an F for the course, or for the assignment. For more details, see the University of Nevada, Reno General Catalog.