Assignment #1
Word Search

Topic(s): C, Eclipse, Writing modular code, scp, gcc
Date assigned: Friday, September 1, 2017
Date due: Friday, September 8, 2017
Points: 15

Problem

I have written most of a C++ program that will find "hidden" words in a rectangular array of letters. The problem is described as follows:

The program will first read the number of rows and the number of columns in the puzzle followed by the puzzle itself (the rectangular array of letters). Next, the program will read the list of words to be searched for in the puzzle into an array of structs.

As output, the program will first echo print all of the data read properly labeled. Next each of the words and whether the word was found or not will be outputted. For each word found, the following information will be printed:

1) word
2) row found
3) column found
4) word's orientation (horizontal, vertical, diagonalLR, diagonalRL)

If the word was not found, print:
1) word
2) the message NOT FOUND

The program assumes that each word occurs at most once (but note that a letter in the puzzle may be part of more than one word).

The program presently searches for words in the following ways:

1) horizontally (left to right)
2) vertically (top to bottom)
3) diagonally (upper-left to lower-right)
4) diagonally (upper-right to lower-left)

It is quite easy to modify the program to search for additional directions

Consider the following puzzle.txt data file:

```
6 4
THAY
EHRJ
```

```
FZEO
YHUB
IJGH
NHIB
THAT
THE
HIP
JOB
```

The program outputs:

```
*****************************
*        Find A Word          *
*****************************

Number of Rows:     6
Number of Columns:  4

Puzzle
------

THAY
EHRJ
FZEO
YHUB
IJGH
NHIB

Words
-----

THAT                            NOT FOUND
THE                             FOUND (Row: 1 Column: 1 Diagonally UL to LR)
HIP                             NOT FOUND
JOB                             FOUND (Row: 2 Column: 4 Vertically)
```

On zeus you will find an almost complete solution to the above described problem with two function prototypes that need to be implemented in C. The mostly completed solution exists in /home/CS300Public/2017 on zeus.

To complete this assignment you must

1. Create an Eclipse project called WordSearch. We will go over how to create a project in class. There needs to be a file called wordsearch.c in the src folder. Create a new folder called data and place the puzzle.txt data in this data file.

2. Write both of the missing functions addBorder and wordSearch.

3. Add search orientations lower right to upper left and lower left to upper right. Once you study the solution given, this should be straight forward.

4. Tar up and submit your solution using the submit script described below. Make sure to extract your tarball and test your solution on zeus before submitting. Test

your solution from the command line. That is how I will be grading and testing your programs.

Additional Notes

1. You must follow the coding standards found on the main CS300 Web page.

2. You must understand ALL of the C code presented in my partial solution, so go through EVERY line and if you have questions, come see me. Any of this code can show up early in some kind of quiz.

3. You will notice that this code is easily modifiable. Keep that going as you write your code. I will run multiple testcases through your final solution and easily modifiable is one such testcase.

4. Your output must look **exactly** like the sample given.

5. Debugging C code can be a challenge. Start early.

How to test your project on zeus

1. Go to where the project WordSearch exists.

2. After cleaning the project, tar up the entire project.
```
punetid@ralph:~> tar czf cs300_1_punetid.tar.gz WordSearch
```

3. Copy the compressed tar file (tarball) to zeus.
```
punetid @ralph:~> scp cs300_1_punetid.tar.gz  punetid@zeus.cs.pacificu.edu:~/Documents/CS300
```

4. Log on the zeus and go to where the tarball is. Extract the tarball.
```
punetid@zeus:~> tar xzf cs300_1_punetid.tar.gz
```

5. Change into the Directory WordSearch/Debug.
```
punetid@zeus:~> cd WordSearch/Debug
```

6. Make the project
```
punetid@zeus:~> make
```

7. Copy the WordSearch executable one level up into the WordSearch directory
```
punetid@zeus:~> cp WordSearch ..
```

8. Move one level up, execute the program, and make sure the output is correct.
```
punetid@zeus:~> cd ..
punetid@zeus:~> ./WordSearch
```

9. Once you are sure your solution is correct, submit the tarball as described next.

How to submit your project

To use the submit script, you must be logged into zeus and be in the directory of where the tarball exists. The command is:

```
submit classname filename
```

EXAMPLE
punetid@zeus:~> submit cs300f17 cs300_1_punetid.tar.gz

will submit the zipped up tar file `cs300_1_punetid.tar.gz`

Once you have successfully submitted a file, you will get a receipt which is a file that will end in **`.receipt`**. You can make sure your file was submitted correctly by typing the command:

```
checkReceipt submittedfile classname receiptfile
```

EXAMPLE
punetid @zeus:~> checkReceipt cs300_1_punetid.tar.gz cs300f17

cs300_1_punetid.tar.gz.cs300f17.receipt

will produce the result:

```
HASH>>>k[illegible]`U[illegible]y[illegible]h
HASH>>>k[illegible]`U[illegible]y[illegible]h[illegible]
SUCCESS! Your receipt is valid
```

DO NOT MOVE OR MODIFY THE SUBMITTED FILE **OR** THE RECEIPT FILE OR THE CHECK RECEIPT COMMAND WILL NOT BE SUCCESSFUL.