CSCI 2240
Program 2

Word Search

Concepts Covered

- 2D Arrays
- strings and chars
- Functions (Recursion?)
- I/O

The assignment is to write a word searching program. The program takes as input an NxN square filled with letters, and a list of words. There are words buried in the square, either left to right, right to left, top to bottom, bottom to top, diagonally, and so on. This is best explained by an example data file and the output. Suppose that this is the input file:

```
T A E D Q Q
Z H P N I U
C K E W D I
V U X O F C
B P I R G K
N R T B R B
EXIT
THE
QUICK
BROWN
FOX
```

Before explaining the program, let me describe the data in more detail. For reading in the square, there is a space after each letter in the square, including the last letter in each row. Looking at the first row, it is "T", then space, then "A", then space, etc. and there is a space and a newline character after the last "Q". By reading the first line of the data, we can tell that this will be a six-by-six square. With respect to the words, they are just a word followed immediately by a newline character. The words will not contain spaces, numbers, etc.

This is a word searching problem. The words such as "EXIT" are contained in the square, oriented in one of eight ways, including left to right, right to left, diagonally down, etc. The output of the program should be the puzzle with the non-used letters removed. Thus, only the words found are printed. The output should be printed exactly like the input square, but with blanks replacing those letters that are never contained within a word.

Here is the output from the above input file:

```
T         Q
  H   N   U
    E W   I
    X O F C
    I R   K
    T B
```

The output must be properly formatted. There should be no blank lines before or after the puzzle and each row of the puzzle is followed by a space and then a newline. So in the example above, there WOULD be a space after the Q in the first row of the solution, there is also a space after the space in the last row.

The square will never be larger than 50 by 50, so you can declare an array of that size or dynamically allocate space. The words will never be longer than the width of the square - six in this example.

The data files to use for the assignment are included in the assignment tar file: data1, data2 and data3; so are the solutions to those puzzles, named solution1, solution2, and solution3.

Make sure to write your program to accept the exact format of the test files.

To test redirect the data file into your program:

```
cat data1 | ./wordsearch
```

or

```
./wordsearch < data1
```

You may redirect the output to a file so that you can compare your output with the provided solutions:

```
./wordsearch < data1 > output1
diff output1 solution1
```

Note: After the programs are collected, any compiled program (executable) in that directory is deleted and the program is recompiled for testing. Specifically, make sure you compile the program with the same options that are used for grading:

```
gcc -Wall –ansi –pedantic wordsearch.c
```

or whatever the correct file name is. If the compiler generates any errors or warnings you will be penalized.

Please include a script file used to compile your program. You can find examples in my lectures posted on Blackboard.