Christian Rodriguez

Algorithms HW 3

1. Problem: Find the location of all words in a matrix
    - Input:
        - m: integer number of rows in a letter matrix
        - n: integer number of columns in a letter matrix
        - w: integer number of words in a list array
    - Other assumed inputs:
        - lettersMatrix: m*n two dimensional array (matrix) of letters
        - wordsList: array of length w
        - solutionMatrix: w*2 matrix of coordinate word solutions
            - These coordinate
    - Output: w x 2 word first letter location (matrix) in the same order as the words list provides
    - Pseudocode:

```
for (i=0; i<words; i++)
    //temp = length of current word wordsList[i]
    for (j=0; j<m; m++)
        for (k=0; k<n; k++)
            SearchRight for the current word[i]
            if SearchRight matches with the current word
                solutionMatrix[i][2] = {m,n}
                end
            SearchUp for the current word[i]
            if SearchUp matches with the current word
                solutionMatrix[i][2] = {m,n}
                end
            SearchLeft for the current word[i]
            if SearchUp matches with the current word
                solutionMatrix[i][2] = {m,n}
                end
            SearchDown for the current word[i]
            if SearchDown matches with the current
            word
                solutionMatrix[i][2] = {m,n}
                end
            end
        end
    if solutionMatrix is still empty
        solutionMatrix[i][2] = {-1,-1}
        end
    end
return solutionMatrix
end
```

2. Prove WordSearch algorithm is correct:
    - My WordSearch algorithm contains a main for loop which iterates through each word in the wordsList we are searching for
    - It also contains two nested for loops which iterate through each element of the lettersMatrix
    - Once inside the double-nested for loop, the algorithm checks each direction (SearchRight/SearchUp/SearchLeft/SearchDown) and compares the string of letters with the word we are searching for
        - We use the length of w to make search specifically for a word of that length
    - Once the double-nested for loop finishes iterating, one last check is performed to determine if we found the previous loops
        - If we could not find the word, we fill the solutionMatrix with garbage coordinates {-1,-1}
    - Once the main loop finishes iterating, solutionMatrix is returned