1. Problem Analysis

A witch is using a spencer boggle word generating device to generates random words for her future spells. The orginal grid could only be a 5 x 5, but after carful considerations she decided to allow for the grid to be any desirable side the person she asks to guess the words wants it to be. The program will either return 1 of two messages, the word has been found or the word has not been found. Words can be found horizontally, vertically, diagonally, and backawards.

|  |  |  |  |
| --- | --- | --- | --- |
| **Known Facts** | **User Requirements** | **Necessary Processing** | **Alternative Solutions** |
| -Grid can be any size the user desires.  -Output found word is word is found  -Output word not found is not found | -Input yes or no to help witch or not  -Input grid dimensions  -Input word to be found  -Input yes or not too find another word or not.  -Input new grid to generate a new grid | -Boolean to check if user will help or not  -string to get user’s answer  -two int variables for the grid dimensions.  -since using getline function throughout have string buffers to convert the grid dimension strings to ints  -do while loop that loops until the user enters a valid answer for y or no  -generate 3 char\*\* variables being for the grid, reset the grid, and generate a new grid if the user desires, and a bool\*\* to mark off used words.  -Generate ASCII numbers to fill the char\*\* variables using double for loop  Do{  -If the user wants a new grid generate it with a double for loop and set the new array values equal to the originally generated array values  -if the word is found from the function call, output a good message and ask if they want to search for another word  -if the word is not found output a sad message and output the grid again to show the user their options.  }while user keeps want to search for words  -delete the dynamically generated arrays  Functions:  Parameters:  -String, char\*\*, bool\*\*, width, height  -double for loop that loops for the first letter of the entered word and then calls the recursion function.  -if the word is found the function returns true  Recursion function: Parameters:  -string, char\*\*, int curRow,int curCol, int curIndex, bool\*\*, int height, int width  -if curIndex is equal to the word length  -if the row is less than 0 or greater than the width minus or column is less than 0 or greater than the height minus one  -if the value in the char\*\* is not equal to the current letter in the string return false  -if the value in the char\*\* is equal to the string array set that value Boolean\*\* which is false  -if the char\*\* is false it means it is visited so return false  -use a Boolean and have multiple recursion calls that can be up, down, right, left, right up, left up, right up, left up | -Boolean to check if user will help or not  -string to get user’s answer  -two int variables for the grid dimensions.  -since using getline function throughout have string buffers to convert the grid dimension strings to ints  -do while loop that loops until the user enters a valid answer for y or no  -generate 4 two dimensional arrays (5 x 5) 3 being chars for the grid, reset the grid, and generate a new grid if the user desires, and one Boolean to mark off used words.  -Generate ASCII numbers to fill the char array variables using double for loop  Do{  -If the user wants a new grid generate it with a double for loop and set the new array values equal to the originally generated array values  -if the word is found from the function call, output a good message and ask if they want to search for another word  -if the word is not found output a sad message and output the grid again to show the user their options.  }while user keeps want to search for words  Functions:  Parameters:  -String, char[5][5], bool[5][5]  -double for loop that loops for the first letter of the entered word and then calls the recursion function.  -if the word is found the function returns true  Recursion function: Parameters:  -string, char[5][5], int curRow,int curCol, int curIndex, bool[5][5],  -if curIndex is equal to the word length  -if the row is less than 0 or greater than the width minus or column is less than 0 or greater than the height minus one  -if the value in the char array is not equal to the current letter in the string return false  -if the value in the char array is equal to the string array set that value Boolean array which is false  -if the char array is false it means it is visited so return false  -use a Boolean and have multiple recursion calls that can be up, down, right, left, right up, left up, right up, left up |

IPO Chart:

|  |  |  |
| --- | --- | --- |
| **Input** | **Processing** | **Output** |
| -Enter Y or N for yes or no to help witch  -Enter grid dimensons  -Enter word to search  -Enter new grid is desired  -Enter if want to search with new grid Y or N  -Enter if want to continue searching for words Y or N | -Two dimensional arrays to store the chars and the booleans  -Double for loop to look for the words  -Function to search for first mathc  -Recusrion function to search for the full word after first letter is matched | -Word is found  -Word is not found |

Psuedo-Code:

Declare variables

Char\*\* arrayLetters

Char\*\* ogArray //a copy of the arrayLetters used to reset arrayLetters after each guess

Char\*\* newArray //array that will hold the newly array if the user wishes to have one

Bool\*\* Booleans //used to mark which letters have been visited

String userEntered //hold the word user searches for

3 booleans used to make sure user enters valid answers

String searchAnother //used to check if user wants to search for another word

Bool outputGrid //used to output the correct grid if a new grid was generated or not

Int randASCII // used to generated random ASCII numbers

Do{

If user does not enter Y and not N

Set the Boolean to false

Output a message with instructions

Else if user does enter y or n

Set Boolean to true

}while Boolean is not true

If user enters Y to help the witch

Two identiacal two while loops (one for row and one for column)

Do{

If user enters number greater than 0

Set Boolean to true

Else

Set Boolean to false

While Boolean is not true

Double for loop

Fills:

//ogArray and arrayLetters with random ASCII’s converted to chars for the grid

Boolean array is filled with false

randASCII = rand()% 25+65 //A to Z is from 65 to 90

arrayLetter = (char) randASCII

orArray = (char) randASCII

output the grid

Do {

Set newGrid initially to false

If(newGrid == false && userEntered == “NEW GRID”)

newGrid =true

outputNewGrid =true

double for loop that generates new grid

randASCII = rand()% 25+65 //A to Z is from 65 to 90

newArray = (char) randASCII

output new grid

//function is the function to search for the first matching letter which then calls a recursion function to search for the full word

else if (newGrid == false && user enters a word to call function and it returns true)

//since Booleans will be replaced with the visited in arrayLetters

double for loop that resets arrayLetters

arrayLetters = ogArray

//if the user enters a word that is not in the grid

else

if(outputNewGrid == false)

double for loop

outputs the array grid that was just generated

else if(outputNewGrid == true)

double for loop

outputs the array grid from newArray and sets arrayLetters values newArray

newGrid =false

}while the user keeps wanting to search for letters

Delete the dynamically generated two dimensional arrays

Else if user does not want to help witch

Output sad message

//function to search for first matching letter

bool wordSearch(string theEntered, char \*\*theArray, bool \*\*theBools, int width, int height)

bool wordFound = false

double for loop that searches for the first matching letter

if first matching letter is found

call recursion function

if word found

return true

return false

//the recursion function

bool theRecruse(string theEntered, char \*\*theArray, int row, int column, int curIndex, bool \*\*theBools, int width, int height )

//base cases

If the current index is equal to the length of the word

Return true //since the word is found

//reutnr false if the the letter found exceeds the row side or the column size

If the row or column is less than 0 or greater than the width-1 or height -1

Return false //since the word searching process exceeds the array dimensons

If the current grid letter is equal to the searching letter

Set the letter in the grid equal to the Boolean array

//this will mark the letter off as visited

If the letter is false

Return false //since it has already been visited.

//recursion calls to search for the full word after the first letter has been matched

Bool isFound = up, down, right left upper right, upper left, lower right, lower left recursion calls

Return bool isFound

**Testing:**

**Test case 1 (prevent user from inputting invalid answers):**

I checked to make sure if my implementation of Booleans to prevent the user from entering invalid answers was working properly by entering invalid answers and then entering valid answers which the code did do.

**Test case 2 (if program matches properly):**

I checked to make sure if my program searches for the word that I enter properly. I checked this entering a word on the grid and it will say it is found if it matches and I entered a word that was not on the grid and the program outputs the word is not found.

**Test case 3 (if new grid outputs properly):**

I checked to make sure if my new grid outputs properly by generating a new grid then trying to match a word on the new grid, entering a wrong word to see if the newly generated grid outputs when the wrong word is inputted.

**Test case 4 (if the program will search outside of the array):**

I checked to make sure if my program tries to search outside of the grid by searching for words that would exceed the array and if the program does exceed it, the program will crash. I know this because it is a slight change to one of the base cases for exceeding the array and I tried the program with and without the small change and with the small change my program did not crash because it prevents it from exceeding the array.

**Test case 5 (If grids are true random grid):**

I ran the program multiple times consecutively to make sure a new random grid outputs every time because I used srand to make sure a new grid will be generated.