Assignment 3
Master of Words

This project will provide practice in C-strings and multidimensional arrays. You will create several functions that are dealing with an array of C-string. Recall that C-strings are character arrays that are null terminated ('\0'). An array of C-string can also be viewed as a 2-dimensional array of characters, or a matrix of characters. For example, let's suppose we have an array of C-string named `WordList` and we declared `MAX_WORD` as 5 and `MAX_WORD_SIZE` as 20. So, we will have a two-dimensional array of chars as `WordList [5][20]`.

```
const int MAX_WORD_SIZE = 20;
const int MAX_WORD = 25;

char WordList[MAX_WORD][MAX_WORD_SIZE];
```

Now, let's say we only stores three words in `WordList`: harry, ron, hermoine. We can imagine `WordList` looking like:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	h	a	r	r	y	\0														
1	r	o	n	\0																
2	h	e	r	m	o	i	n	e	\0											
3																				
4																				

Observe that the rows in this matrix are the words and the columns are the characters of the words. You may assume that words will be no more than 19 characters in length (+1 for the null character) and that words stored in the wordList contain no white space. The following call:

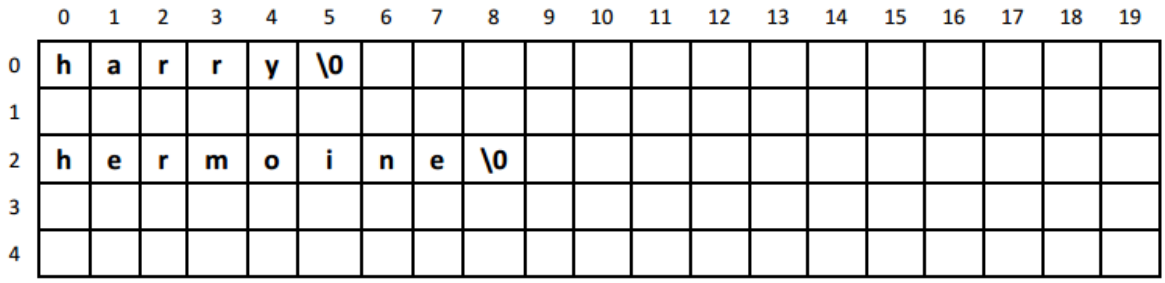
```
cout << WordList[0][2];
```

will output the third character in the first word, or the first 'r' in harry. Also,

```
cout << WordList[1];
```

will output the entire second word or "ron". Note that empty cstrings (C-strings with just the null character as their first element) will not be considered words for this project. As you see in the example above, not all rows in the two dimensional array is always full. If you reach an empty string, it means you reach the end of the sequence of words in the array.

Following this, there should be no empty strings between words in the list. For example, the following is an invalid list and would be considered incorrect. In order to be valid, hermoine would have to be moved to the second row. Or, else you may consider "harry" to be the only word in the `wordList`. This means that the empty string in the row 1 indicates the end of the sequence of words in `wordList` regardless of the size of the array (`MAX_WORD`).



You may not use or include the string library, or any other libraries not already included. However, you are provided with and may use the functions in the C-string library, e.g. `strlen`, `strcpy`, `strcmp`, etc. You may create your own helpers functions and are encouraged to do so. We will only test the functions described in this specification.

What you get

You will be provided with a file in addition to this specification named wordlist.cpp. This file will contain your function implementations, along with any helper functions prototype/definitions. You cannot change the function signatures for the functions described in this specification. You may not add to global constants.

When you setup your environment with this file, make sure everything compiles without error before making any changes. You can modify main function as you wish. When grading, we will ignore your main function.

Meet the functions

You are to implement ten functions with following specifications. Each function will have a difficulty rating from 1 to 3, 1 being easiest. The number of points each function is worth, are consistent with its difficulty rating. Those points will be divided among the different test cases we will use to validate your implementations. An easy function may take anywhere from 10 to 30 minutes to fully implement and a hard function may take anywhere from 1 to 3 hours. Below is a brief table of functions to be implemented, detailed specification is below.

Function	Rating	Function	Rating
findWord	1	printList	1
removeWord	1	removeDuplicates	3
removeAllInstances	2	findWordSequence	3
insertWord	2	findMinimum	3
initialize	3	numOfWords	1

Function1: `void printList(char words[][MAX_WORD_SIZE]);`

Difficulty: 1
Specification: it prints the list in `Words` to the console with a space between each word and a newline after the last; there is no space after the last word.
Usage:

```
char list[15][MAX_WORD_SIZE] = { "tom", "jerry", "nibbles" };
printList(list); // print “tom jerry nibbles\n” to console
```

Function 2: `int numOfWords(const char words[][MAX_WORD_SIZE]);`

Difficulty: 1
Specification: It returns the number of words in the array `words`. Note that it would be different from `MAX_WORD` and you have to count non-empty C-strings in the array.
Usage:

```
char list[15][MAX_WORD_SIZE] = { "I", "think", "I", "will", "be", "there" };
cout << numOfWords(list); // print 6
```

Function3: `void initialize(char words[][MAX_WORD_SIZE], const char str[], char delimiter);`

Difficulty: 3
Specification: It splits the C-string `str` by the char `delimiter`.
Usage:

```
char str[100] = "I think I will be there at 5";
char words[15][MAX_WORD_SIZE];
initialize(words, str, ' '); // words will initialize to { "I", "think", "I" ,
                             //   "will", "be", "there", "at", "5"}
```

Function 4: `int findWord(char words[][MAX_WORD_SIZE], const char str[]);`

Difficulty: 1
Specification: Finds the first occurrence of `str` in `words` and return the index. If not found, return -1.
Usage:

```
char list[15][MAX_WORD_SIZE] = { "harry", "ron", "hermione" };
cout << findWord(list, "ron");    //print 1
cout << findWord(list, "harry");  //print 0
```

Function 5: `int removeWord(char words[][MAX_WORD_SIZE], unsigned int i);`

Difficulty: 1
Specification: Removes i^{th} word in the array `words`. If the i is larger than the index of the last word, simply remove the last word in `words`. If the `words` is empty, return -1.
Usage:

```
char list[15][MAX_WORD_SIZE] = { "hello!" , "how", "are" , "you?" };
removeWord(list, 1);
printList(list);     // print hello! are you?
removeWord(list, 5);
printList(list); // print hello! are
```

Function 6: `void findWordSequence(char words1[][MAX_WORD_SIZE], const char words2[][MAX_WORD_SIZE], int& start, int& end);`

Difficulty: 3
Specification: It tries to locate every word in `words2` inside `words1` and return the start and end indices of subsequence in `words1` which has `words2`. If the subsequence does not exist, it will set start and end to -1;
Usage:

```
char list1[15][MAX_WORD_SIZE] =
       { "I", "think", "I" , "will", "be" , "there", "at", "5"};
char list2[3][MAX_WORD_SIZE] = { "think", "be", "at" };
int start , end;
findWordSequence(list1, list2, start, end);
cout << start << " " << end << endl; // print 1 6
```

Function 7: `int insertWord(char words[][MAX_WORD_SIZE],const char new_word[], int i);`

Difficulty: 2

Specification: It inserts `new_word` in position i of array words. If i is larger than the last word's index, it will append the `new_word` to the end of array words. If the array is full it will return -1, indicating that it could not insert the word.

Usage:

```
char list[8][MAX_WORD_SIZE] = { "tom", "jerry", "nibbles" };
insertWord(list, "harry" , 1);
printList(list); // print tom harry jerry nibbles
insertWord(list, "ron", 7);
printList(list); // print tom harry jerry nibbles ron
cout << findWord("ron"); // print 4
```

Function 8: `char* findMinimum(char words[][MAX_WORD_SIZE]);`

Difficulty: 3

Specification: It returns the pointer to the word which is lexicographically minimal. Lexicographical order of a sequence of words is the same order as they appear in a dictionary, meaning that words starting with a will be before those starting with b. In this part, you can assume that `words` will only consist of alphabets. Note that it will compare the words case-insensitive. For example:

```
"apple"  is less than "Zoo"
"animal" is less than "ANT"
"ANiMaL" is less than "ferry"
```

Usage:

```
char list[15][MAX_WORD_SIZE] = { "I", "think", "be", "will", "be", "there" };
char* str = findMinimum(list);
cout << str; // return pointer to the third element (first occurrence of "be")
```

Function 9: `int removeDuplicates(char words[][MAX_WORD_SIZE]);`

Difficulty: 3

Specification: Searches through `words` and removes all duplications of a word, keeping only the first occurrence of the word. Returns the number of words removed. `removeDuplicates` should preserve the order of the remaining words.

Usage:

```
char list[15][MAX_WORD_SIZE] =
{"Perdido","City","Scar","Council","Scar","Scar","City","Embassytown"};
int retval = removeDuplicates(list);
printList(list);     // prints "Perdido City Scar Council Embassytown\n"
cout << retval;      // prints 3
```

Function 10: `int removeAllInstances(char words[][MAX_WORD_SIZE], const char str[]);`

Diffuculty: 2
Specification: Deletes all instances of the word `str` from array of C-string `words`. It will return the number of elements removed.
Usage:

```
char list[15][MAX_WORD_SIZE] =
        { "Vahab" , "Nazanin", "Behnam" , "Nazanin", "Sepideh", "Sharath"};
int count;
count = removeAllInstances(list, "Nazanin");
cout << count; // print 2
printList(list); // Vahab Behnam Sepideh Sharath
```

Recommendations:

- It is good practice to develop your test cases prior to developing your code. Identify, the different inputs to your functions that will exercise a single aspect of that function and have a known expected output. That way when you implement that particular aspect you can see if it behaves as expected, if not you make the necessary adjustments.
- Implement `initialize`, `numOfWords` and `printList` first. Those are the basic functions needed to test your code.
- Work on a single function at a time, after implementing initialize, `numOfWords` and `printList` work on the easier functions then move onto the harder functions. You should notice that many of the functions share similarities in their implementations with other functions. Take advantage of this to save yourself time.
- You may call the functions you implemented in the implementation of other functions.

What to turn in:

Guidelines on where to submit the project will be available 48 hours before the assignment.

Your submission file must be in the following format: the completed wordlist.cpp must be compressed into a single .zip file. The ZIP file name must be in the following format:

[StudentID#]_proj3.zip

For instance if my student ID is 123456789 and I am submitting my solution for assignment 3, then I am going to compress wordlist.cpp file and rename the zip file to: 123456789_proj3.zip

You should only include the wordlist.cpp file that you modified into the zip file. Do not submit/include any other file such as the executable or the contents of the debug folder.

Same as other assignments, a good sanity check is to check your zip file for corruption by extracting it and testing whether it did compress it successfully.