ENGG1110 Problem Solving by Programming (2018-2019 Term 1)
Project – Word Search Puzzle

1. Introduction

A Word Search Puzzle is a classic game to find words hidden in a **letter grid**, where all the words are provided in a **word list**.

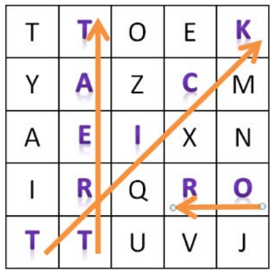
The letter grid consists of some cells organized in a square or a rectangle shape, where each cell contains a letter. An example is shown as follows:

T	T	O	E	K
Y	A	Z	C	M
A	E	I	X	N
I	R	Q	R	O
T	T	U	V	J

The word list contains some words to be found in the letter grid. An example is shown as follows:

TRICK
OR
TREAT

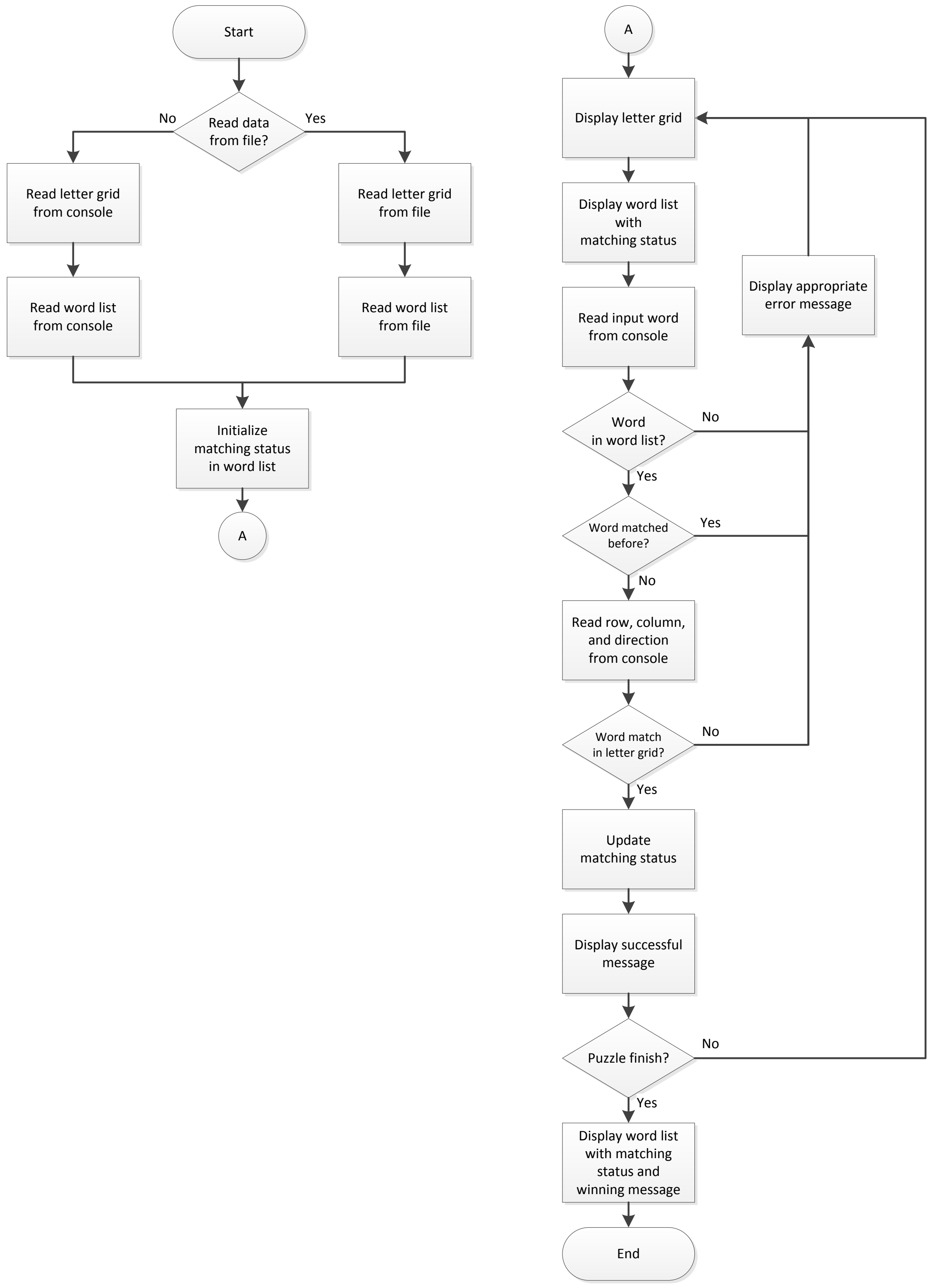
The words in the word list can be found in one of the 8 directions in the letter grid, i.e., 2 for horizontal (i.e., left to right or right to left), 2 for vertical, and 4 for diagonal. An example is shown as follows:



In this project, your task is to use **Code::Blocks** in **Windows** to create a Console Application of a Word Search Puzzle using C language.

2. Program Design

A skeleton code (in **SkeletonCode.c**) is given and you are required to complete it using **Code::Blocks** in this project. The program flow is shown as follows:



Start
Read data from file?
No
Yes
Read letter grid from console
Read word list from console
Read letter grid from file
Read word list from file
Initialize matching status in word list
A
A
Display letter grid
Display word list with matching status
Read input word from console
Word in word list?
No
Yes
Word matched before?
Yes
No
Read row, column, and direction from console
Word match in letter grid?
No
Yes
Update matching status
Display successful message
Puzzle finish?
No
Yes
Display word list with matching status and winning message
End
Display appropriate error message

3. Schedule

The following shows the suggested schedule, the code submission deadline, and the date for the project demo and presentation:

Week	Date	Tasks
9	Oct 31	• Choose Data Source: Console or File? (Section 4.2) • Read Data from Console (Section 4.3) • Initialize Matching Status (Section 4.5) • Display Letter Grid and Word List (Section 4.6) By the end of this week, your program should be able to: ○ Print the letter grid and the word list that are read from the console
10	Nov 7	• Read Data from File (Section 4.4) • Read Input Word from Console and Check Word List (Section 4.7) By the end of this week, your program should be able to: ○ Print the letter grid and the word list that are read from the file ○ Read an input word from the user and decide whether it is in the word list or matched before
11	Nov 14	• Read Row, Column, and Direction from Console (Section 4.8) • Check Letter Grid (Section 4.9) • Check Winning Condition (Section 4.10) By the end of this week, your program should be able to: ○ Allow a player to finish the whole puzzle
12	Nov 21	• Print Secret Table (Section 5) • Final Testing on Lab Computer By the end of this week, your program should be able to: ○ Allow a player to enter the secret code to print the secret table ○ Run smoothly on a lab computer, which will be used in the project demo and presentation
13	**Nov 26**	• Project Submission (by 23:59 on Blackboard)
	Nov 28	• Project Demo and Presentation (in the lesson)

4. Detailed Design

4.1 Macros and Variable Declarations

Suppose the macro **MAX_GRID** is defined in the beginning of the source code:

```
#define MAX_GRID 10
```

And we have the following in the other part of the source code:

```
char letterGrid[MAX_GRID][MAX_GRID]
```

The above will be replaced by

```
char letterGrid[10][10]
```

automatically before the source code compiles.

The following table shows the macros defined in the beginning of the skeleton code. They are mainly used for the declarations of arrays and parameters.

Macro	Value	Explanation
MAX_GRID	10	Maximum size of the letter grid, which is a square The value is no larger than 10
MAX_WORDLIST	8	Maximum size of the word list
MAX_WORD	11	Maximum number of bytes of a word in the word list and a word inputted by the user, including the terminating NULL character The value is no larger than MAX_GRID + 1
MAX_FILENAME	260	Maximum number of bytes in a filename, including the terminating NULL character

Your program should run correctly even if the macro values are changed reasonably.

Besides, you cannot use any global variables in this project. In other words, all variables must be declared inside functions.

4.2 Choose Data Source: Console or File?

The program will first ask whether to read the letter grid and the word list from files with the following prompt:

```
Read data from file [Y/N]?
```

You can assume that the user must input either Y or N. You are required to implement this feature in the **main** function.

4.3 Read Data from Console

If the user did not choose to read the data from file, the program will simply read the letter grid and the word list from the console. The following two functions have already been implemented in the skeleton code. You are required to study them first and then invoke them from the **main** function:

- `int readLetterGridFromConsole(char letterGrid[MAX_GRID][MAX_GRID])`
    - Read the letter grid from console into the 2D array parameter **letterGird**
    - Return the size of the letter grid
- `int readWordListFromConsole(char wordList[MAX_WORDLIST][MAX_WORD])`
    - Read the word list from console into the 2D array parameter **wordList**
    - Return the size of the word list

The program will first ask the user for the size of the letter grid with the following prompt:

```
Enter the size of the letter grid:
```

You can assume that the input must be an integer of at least 2 but no larger than **MAX_GRID**. Let the input be **gridSize**. Then, the program will ask the user for the letter grid with the following prompt:

```
Enter the letter grid:
```

You can assume that the input must be **gridSize** lines of uppercase letters, where the number of letters on each line is **gridSize**. An example is shown as follows (underlined characters in blue are user inputs):

```
Enter the size of the letter grid:
5
Enter the letter grid:
CTOEK
YAZCM
AEIXN
IRQRO
TTUVJ
```

You can assume that:

1. **gridSize** is at least 2 but no larger than **MAX_GRID**.
2. All letters are uppercase letters.

After reading the letter grid, the program will ask the user for the size of the word list with the following prompt:

```
Enter the size of the word list:
```

You can assume that the input must be an integer of at least 1 but no larger than **MAX_WORDLIST**. Let the input be **listSize**. Then, the program will ask the user for the word list with the following prompt:

```
Enter the word list:
```

You can assume that:

1. **listSize** is at least 1 but no larger than **MAX_WORDLIST**.
2. The length of each word is at least 2 but no longer than **MAX_WORD – 1**.
3. There are no duplicate words in the word list.
4. Each word must appear once and only once in the letter grid in one of the 8 directions.
5. All letters are uppercase letters.

An example is shown as follows (underlined characters in blue are user inputs):

```
Enter the size of the word list:
3
Enter the word list:
TRICK
OR
TREAT
```

After reading the word list, we have **listSize** = 3 and

wordList[0] is “TRICK”
wordList[1] is “OR”
wordList[2] is “TREAT”

4.4 Read Data from File

If the user chose to read the data from file, the program will ask for two filenames and then read the letter grid and the word list from the specified files. You are required to complete and then invoke the following two functions from the **main** function:

- `int readLetterGridFromFile(char letterGrid[MAX_GRID][MAX_GRID])`
  - Read the letter grid from file into the 2D array parameter **letterGird**
  - Return
    - the size of the letter grid if successful
    - -1 if there is any file reading error
- `int readWordListFromFile(char wordList[MAX_WORDLIST][MAX_WORD])`
  - Read the word list from file into the 2D array parameter **wordList**
  - Return
    - the size of the word list if successful
    - -1 if there is any file reading error

The program will first ask the user for the filename of the letter grid with the following prompt:

```
Enter filename of the letter grid:
```

The file format is described as follows. The first number indicates the size of the grid, denoted as **gridSize**. Then, there are **gridSize** lines of uppercase letters, where the number of letters on each line is **gridSize**. An example is shown as follows:

LetterGrid.txt
5 CTOEK YAZCM AEIXN IRQRO TTUVJ

You can assume that:

1. The length of the filename input by the user is no longer than **MAX_FILENAME – 1**.
2. The file format must be correct.
3. **gridSize** is at least 2 but no larger than **MAX_GRID**.
4. All letters are uppercase letters.

If the file cannot be found or there is any IO error, the program will display the following message and terminate.

```
Error in reading the letter grid file. Program terminates.
```

After reading the letter grid from the file, the program will ask the user for the filename of the word list with the following prompt:

```
Enter filename of the word list:
```

The file format is described as follows. The first number indicates the size of the list, denoted as **listSize**. Then, there are **listSize** words on each line. An example is shown as follows:

WordList.txt
3 TRICK OR TREAT

You can assume that:

1. The length of the filename input by the user is no longer than **MAX_FILENAME – 1**.
2. The file format must be correct.
3. **listSize** is at least 1 but no larger than **MAX_WORDLIST**.
4. The length of each word is at least 2 but no longer than **MAX_WORD – 1**.
5. There are no duplicate words in the word list.
6. Each word must appear once and only once in the letter grid in one of the 8 directions.
7. All letters are uppercase letters.

After reading the word list, we have **listSize** = 3 and

wordList[0] is "TRICK"
wordList[1] is "OR"
wordList[2] is "TREAT"

If the file cannot be found or there is any IO error, the program will display the following message and terminate.

```
Error in reading the word list file. Program terminates.
```

4.5 Initialize Matching Status

After reading the letter grid and the word list from the console or the files, you are required to initialize the matching status of each word in the word list in the **main** function. In the skeleton code, the matching status is represented by the following integer array:

```
int matchingStatus[MAX_WORDLIST];
```

If an array element is 0, the corresponding word has not been matched. If an array element is 1, the corresponding word has already been matched. For example, suppose **listSize** is 3 and we have:

wordList[0] is “TRICK”	**matchingStatus[0]** is 0
wordList[1] is “OR”	**matchingStatus[1]** is 0
wordList[2] is “TREAT”	**matchingStatus[2]** is 1

That means the word “TREAT” has been matched while the words “TRICK” and “OR” have not.

Before the game starts, all the elements from **matchingStatus[0]** to **matchingStatus[listSize – 1]** are initialized to 0. When a word is matched later, the corresponding **matchingStatus** element will be set to 1.

4.6 Display Letter Grid and Word List

After initializing the matching status of the word list, the program will display the puzzle on the screen. You are required to complete and then invoke the following two functions from the **main** function:

- `void printLetterGrid(char letterGrid[MAX_GRID][MAX_GRID], int gridSize)`
    - Display the letter grid stored in **letterGrid** of size **gridSize**

- `void printWordList(char wordList[MAX_WORDLIST][MAX_WORD], int listSize, int matchingStatus[MAX_WORDLIST])`
    - Display the word list stored in **wordList** of size **listSize** with the matching status for each word (stored in **`matchingStatus`**)

An example is shown as follows, where all three words are not matched initially:

```
### 5 x 5 Letter Grid ###
  + 0 1 2 3 4
+ + + + + + +
0 + T T O E K
1 + Y A Z C M
2 + A E I X N
3 + I R Q R O
4 + T T U V J
### Word List of Size 3 ###
[ ] TRICK
[ ] OR
[ ] TREAT
```

Suppose the word "TREAT" is matched later, the following will be displayed:

```
### 5 x 5 Letter Grid ###
  + 0 1 2 3 4
+ + + + + + +
0 + T T O E K
1 + Y A Z C M
2 + A E I X N
3 + I R Q R O
4 + T T U V J
### Word List of Size 3 ###
[ ] TRICK
[ ] OR
[X] TREAT
```

4.7 Read Input Word from Console and Check Word List

After displaying the puzzle, the program will ask the user for the word to be matched with the following prompt:

```
Enter the word:
```

You can assume that the input must be a non-empty string no longer than **MAX_WORD – 1**. After reading the input word, you are required to complete and then invoke the following function from the **main** function:

- **`int checkWordList(char wordList[MAX_WORDLIST][MAX_WORD], int listSize,`**
  `                    int matchingStatus[MAX_WORDLIST], char inputWord[MAX_WORD])`
  - Check whether **inputWord** is in **wordList** of size **listSize** and whether it is matched before
  - Return
    - the index of **inputWord** in **wordList** (i.e., a value between 0 and **listSize – 1** inclusively) if **inputWord** is in **wordList** and has not been matched before
    - **listSize** if **inputWord** is in **`wordList`** but has been matched already
    - -1 if **inputWord** is not in **`wordList`**

If the input word is not in the word list, the program will display the following message:

```
The input word is not in the word list.
```

and then display the prompt again for another user input.

If the input word is in the word list but has already been matched before, the program will display the following message:

```
The input word has already been matched before.
```

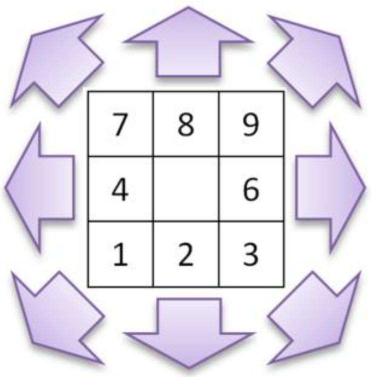
and then display the prompt again for another user input.

4.8 Read Row, Column, and Direction from Console

After reading an input word in the word list that has not been matched before, the program will ask the user for the row number, the column number, and the direction by displaying the following prompt:

```
Enter the row number, the column number, and the direction (1-4 or 6-9):
```

- Row number
    - You can assume that the input must be an integer of a valid row number.
- Column number
    - You can assume that the input must be an integer of a valid column number.
- Direction
    - You can assume that the user must input an integer in the range of 1-4 or 6-9.
    - The mapping of the input integer and the direction is shown as follows:



Examples for the matching are shown in the next subsection. You are required to implement the above in the **main** function.

4.9 Check Letter Grid

After reading the user input, the program will try to match the input word in the specified location and direction in the letter grid. You are required to complete and then invoke the following function from the **main** function:

-
  ```
  int checkLetterGrid(char letterGrid[MAX_GRID][MAX_GRID], int gridSize,
                      char inputWord[MAX_WORD],
                      int matchRow, int matchCol, int matchDirection)
  ```
    - Check whether **inputWord** is in **letterGrid** in the specified **matchRow**, **matchCol** and **matchDirection**
    - Return
        - 1 if **inputWord** is in **letterGrid** in the specified **matchRow**, **matchCol** and **matchDirection**
        - 0 if **inputWord** cannot be found in **letterGrid** in the specified **matchRow**, **matchCol** and **matchDirection**
        - -1 if the search of **inputWord** exceeds the boundary of the **letterGrid**
    - Example:

```
  + 0 1 2 3 4
+ + + + + + +
0 + T T O E K
1 + Y A Z C M
2 + A E I X N
3 + I R Q R O
4 + T T U V J
```

inputWord	matchRow	matchCol	matchDirection	RETURN VALUE
TRICK	4	0	9 (top-right)	1 (word found)
TREAT	4	0	9 (top-right)	0 (word not found)
TRICK	4	0	1 (bottom-left)	-1 (search exceed boundary)

You may need to invoke this function to print the secret table (see later for details). Hence, it is not recommended for this function to print anything (except debugging messages, if any).

After invoking the above function from the **main** function, the latter will check the return value, update the corresponding **matchingStatus** element to 1 (only if the word is found), and display one of the following messages:

```
The word is found!
```

```
The input word cannot be found in the given location.
```

```
The search exceeds the boundary of the letter grid.
```

4.10 Check Winning Condition

After matching a word, the program will check whether all the words in the word list are matched:

- If no, go back to "Display Letter Grid and Word List (Section 4.6)" and continue the puzzle.
- If yes, the program will display the word list with a winning message and then terminate. An example is shown as follows:

```
### Word List of Size 3 ###
[X] TRICK
[X] OR
[X] TREAT
You have finished the puzzle.
Congratulations!
```

5. Secret Table

When the program prompts for the input word and the user enters SECTBL, a secret table will be shown. The secret table contains the correct answer (row, column, and direction) for each word in the word list. After that, the program will prompt for the input word again.

An example is shown as follows (underlined characters in blue are user inputs):

```
### 5 x 5 Letter Grid ###
  + 0 1 2 3 4
+ + + + + + +
0 + T T O E K
1 + Y A Z C M
2 + A E I X N
3 + I R Q R O
4 + T T U V J
### Word List of Size 3 ###
[ ] TRICK
[ ] OR
[X] TREAT
Enter the word:
SECTBL
### Secret Table ###
TRICK: R4 C0 D9
OR: R3 C4 D4
TREAT: R4 C1 D8
Enter the word:
```

You are not allowed to accept extra input files to get the answers. Instead, please design your own algorithm to find the answers automatically. (Hint: you may find the **checkLetterGrid()** function useful.)

You are required to implement your algorithm in the following function and then invoke it in an appropriate location:

- **`void printSecretTable(/* put your parameter list here*/)`**

For the size of the array parameter, you are required to use the macro (in Section 4.1) instead of hard coding the value.

You can assume that:

1. The word list does not contain the word SECTBL.
2. **MAX_WORD** is at least 7 (= length of SECTBL + the terminating NULL character).

Note that the flow chart in "Program Design" does not include the secret table.

6. Sample Run

A sample run is shown as follows (underlined characters in blue are user inputs) for your reference. Blank lines are added to improve the readability, it is not necessary for your program to include them.

```
Read data from file [Y/N]?
N
Enter the size of the letter grid:
10
Enter the letter grid:
OSAMPLEOOO
OOOOOOOOOO
OOOOOOOOOO
OOOOOOOOOO
OOOOOOOOOO
OOOOOOOOOO
OOOOOOOOOO
OOOOOOOOOO
OOOOOOOOOO
OOOOOOONUR
Enter the size of the word list:
2
Enter the word list:
SAMPLE
RUN


### 10 x 10 Letter Grid ###
  + 0 1 2 3 4 5 6 7 8 9
+ + + + + + + + + + + +
0 + O S A M P L E O O O
1 + O O O O O O O O O O
2 + O O O O O O O O O O
3 + O O O O O O O O O O
4 + O O O O O O O O O O
5 + O O O O O O O O O O
6 + O O O O O O O O O O
7 + O O O O O O O O O O
8 + O O O O O O O O O O
9 + O O O O O O O N U R
### Word List of Size 2 ###
[ ] SAMPLE
[ ] RUN
Enter the word:
SAMPLE
Enter the row number, the column number, and the direction (1-4 or 6-9):
0 1 6
The word is found!
```

```
### 10 x 10 Letter Grid ###
  + 0 1 2 3 4 5 6 7 8 9
+ + + + + + + + + + + +
0 + O S A M P L E O O O
1 + O O O O O O O O O O
2 + O O O O O O O O O O
3 + O O O O O O O O O O
4 + O O O O O O O O O O
5 + O O O O O O O O O O
6 + O O O O O O O O O O
7 + O O O O O O O O O O
8 + O O O O O O O O O O
9 + O O O O O O O N U R
### Word List of Size 2 ###
[X] SAMPLE
[ ] RUN
Enter the word:
SAMPLE
The input word has been matched already.


### 10 x 10 Letter Grid ###
  + 0 1 2 3 4 5 6 7 8 9
+ + + + + + + + + + + +
0 + O S A M P L E O O O
1 + O O O O O O O O O O
2 + O O O O O O O O O O
3 + O O O O O O O O O O
4 + O O O O O O O O O O
5 + O O O O O O O O O O
6 + O O O O O O O O O O
7 + O O O O O O O O O O
8 + O O O O O O O O O O
9 + O O O O O O O N U R
### Word List of Size 2 ###
[X] SAMPLE
[ ] RUN
Enter the word:
RU
The input word is not in the word list.


### 10 x 10 Letter Grid ###
  + 0 1 2 3 4 5 6 7 8 9
+ + + + + + + + + + + +
0 + O S A M P L E O O O
1 + O O O O O O O O O O
2 + O O O O O O O O O O
3 + O O O O O O O O O O
4 + O O O O O O O O O O
5 + O O O O O O O O O O
6 + O O O O O O O O O O
7 + O O O O O O O O O O
8 + O O O O O O O O O O
9 + O O O O O O O N U R
### Word List of Size 2 ###
[X] SAMPLE
[ ] RUN
Enter the word:
RUNE
The input word is not in the word list.
```

```
### 10 x 10 Letter Grid ###
  + 0 1 2 3 4 5 6 7 8 9
+ + + + + + + + + + + +
0 + O S A M P L E O O O
1 + O O O O O O O O O O
2 + O O O O O O O O O O
3 + O O O O O O O O O O
4 + O O O O O O O O O O
5 + O O O O O O O O O O
6 + O O O O O O O O O O
7 + O O O O O O O O O O
8 + O O O O O O O O O O
9 + O O O O O O O N U R
### Word List of Size 2 ###
[X] SAMPLE
[ ] RUN
Enter the word:
RUN
Enter the row number, the column number, and the direction (1-4 or 6-9):
9 7 6
The input word cannot be found in the given location.


### 10 x 10 Letter Grid ###
  + 0 1 2 3 4 5 6 7 8 9
+ + + + + + + + + + + +
0 + O S A M P L E O O O
1 + O O O O O O O O O O
2 + O O O O O O O O O O
3 + O O O O O O O O O O
4 + O O O O O O O O O O
5 + O O O O O O O O O O
6 + O O O O O O O O O O
7 + O O O O O O O O O O
8 + O O O O O O O O O O
9 + O O O O O O O N U R
### Word List of Size 2 ###
[X] SAMPLE
[ ] RUN
Enter the word:
RUN
Enter the row number, the column number, and the direction (1-4 or 6-9):
9 9 6
The search exceeds the boundary of the letter grid.
```

```
### 10 x 10 Letter Grid ###
  + 0 1 2 3 4 5 6 7 8 9
+ + + + + + + + + + + +
0 + O S A M P L E O O O
1 + O O O O O O O O O O
2 + O O O O O O O O O O
3 + O O O O O O O O O O
4 + O O O O O O O O O O
5 + O O O O O O O O O O
6 + O O O O O O O O O O
7 + O O O O O O O O O O
8 + O O O O O O O O O O
9 + O O O O O O O N U R
### Word List of Size 2 ###
[X] SAMPLE
[ ] RUN
Enter the word:
SECTBL
### Secret Table ###
SAMPLE: R0 C1 D6
RUN: R9 C9 D4
Enter the word:
RUN
Enter the row number, the column number, and the direction (1-4 or 6-9):
9 9 4
The word is found!
### Word List of Size 2 ###
[X] SAMPLE
[X] RUN
You have finished the puzzle.
Congratulations!
```

7. Academic Honesty and Declaration Statement

Attention is drawn to University policy and regulations on honesty in academic work, and to the disciplinary guidelines and procedures applicable to breaches of such policy and regulations. Details may be found at https://www.cuhk.edu.hk/policy/academichonesty/.

Please place the following declaration statement as the comment in the beginning of your .c source code and fill in your information.

```
/**
 * ENGG1110 Problem Solving by Programming
 *
 * Course Project
 *
 * I declare that the project here submitted is original
 * except for source material explicitly acknowledged,
 * and that the same or closely related material has not been
 * previously submitted for another course.
 * I also acknowledge that I am aware of University policy and
 * regulations on honesty in academic work, and of the disciplinary
 * guidelines and procedures applicable to breaches of such
 * policy and regulations, as contained in the website.
 *
 * University Guideline on Academic Honesty:
 *   https://www.cuhk.edu.hk/policy/academichonesty/
 *
 * Student Name  : <your name>
 * Student ID    : <your student ID>
 * Class/Section : <your class/section>
 * Date          : <date>
 */
```

8. Testing Platform

Your submission will be graded by using **Code::Blocks** in **Windows**. Please note that there may be some problems in opening a project in Windows if the project is created in other operating systems, such as macOS and Linux.

9. Submission

The deadline of the project submission is **Nov 26 (Mon) 23:59**. Please follow the following steps to submit your work.

1. Compress your **whole Code::Blocks project folder** into a file in ZIP format named as: ENGG1110<your class/section>_<your student ID>.zip (E.g., ENGG1110G_1155012345.zip)
2. Visit Blackboard for CUHK and login with your OnePass (CWEM) password.
3. Visit the page for ENGG1110 and go to Project → Project Submission.
4. Upload and submit your file prepared in Step 1.
5. Download your submission from Blackboard to see if it can be extracted and then opened by Code::Blocks in Windows successfully.

Resubmissions are allowed. But only the latest one will be graded. 10% of the project marks will be deducted for late submissions within one week (i.e., by Dec 3 (Mon) 23:59). Late submissions more than one week will not be graded.

10. Project Demo and Presentation

The project demo and presentation is scheduled on **Nov 28 (Wed) in the lesson**. It is conducted in **English**. Each student will need to:

1. Compile and execute the program on a lab computer using **Code::Blocks** in **Windows**.
2. Explain the code and the program design

This part contributes 10% of the project marks. Marks for this part will only be given during the project demo and presentation. Even if your project is incomplete or you plan to submit the project late, you should still come and try to get some marks. However, a student who is absent on that day will get **0 marks** for this part.

- END -