APS 105 — Computer Fundamentals

Lab #6: 2D Arrays

Winter 2022

You must use `examify.ca` to electronically submit your solution by 11:59 pm on **Saturday, March 12, 2022**. Please note that in this lab, you are only asked to implement and submit functions. However, to test your functions, you will need to write a complete `C` program.

Objective

The goal of this laboratory is to solve complex problems involving the usage of 2D arrays.

Preparation

Read through this entire document carefully, and do the work to create the programs that are described below. You are encouraged to:

- Read the class bulletin board on Piazza, to see if others had similar problems. If you do not see anything helpful there, ask a question. Do not ask for, or ever give a full or partial solution to the lab assignment.
- During the lab, ask for assistance from the lab TA assigned to you in your lab subsection.

Start and follow the suggested steps below:

- Write your pseudocode that is in a plain language describing and planning the steps in solving the problem at hand.
- Follow up your plan by writing a code on paper, i.e., stay away from typing on your computer at this time.
- Try to review your code and identify any mistakes that you can identify, i.e., walk through your code.
- Use your computer, type and run your code.
- Run the test cases provided to ensure proper functionality of your code.
- Think if there are corner case by developing yourself difficult examples. Solve the examples you developed by hand and check if your code is giving the expected outcome.

Part 1 — Word search puzzle

The twins Kartik and Kunaal want to play word search puzzles with their class. They want to be faster than everyone else in their class, so they decided to write a `C` code to help them find words with a click of a button. Little did they know, everyone in class decided to do this question.

You are required to implement `search2D` function.

```
void search2D(char word[], int wordSize, const int Size, char grid[Size][Size]);
```

`search2D` receives the word to look for `char word[]`, the size of the word `int wordSize`, size of the grid `const int Size` and the puzzle grid `char grid[Size][Size]`. The word size can be between 2 and 23. You are NOT required to check for the validity of the word size.

`search2D` should look for the first letter in the word *row by row*. If it finds the first letter of the word in the puzzle, it should look for the remaining of the word in the 8 directions from the south and goes clockwise, i.e. south, south-west, west, north-west, north, north-east, east and lastly south-east. If the word is found, the function should print the row and column location of the first letter of the word and the direction of the word from that location. Row and column count starts from 0.

These are several tasks in one function; therefore you should implement these two helper functions that `search2D` will call.

```
bool search1D(char word[], int wordSize, const int Size, char grid[Size][Size],
 int row, int col, int rowDir, int colDir);

 void printFoundLocation(int rowDir, int colDir);
```

`search1D` should receive the word in `char word[]`, word size in `int wordSize`, size of grid in `const int Size` and the puzzle grid in `char grid[Size][Size]`, the row and column indices in the grid of the first letter in the word in `int row` and `int col`, and the row and column direction to look in `int rowDir` and `int colDir`. The function should return if it found the word in this direction or not.

`printFoundLocation` should receive the direction in `int rowDir` and `int colDir` and prints the direction equivalent to these two integers.

You are encouraged to use more functions in your code.

Although you are asked to only implement three functions, you will have to implement the entire program to test your code on VS Code. We suggest you test your code with a small 2D array, example 4 by 4 or 5 by 5 and small words of 2 or 3 letters.

In the example outputs below, please note that your output is in bold.

In this example, Ben is there at row 0 and column 0 in the south and south-east direction, but the code will only output south, as it checks in the order mentioned above.

```
The word search puzzle is
B Q I F J Q K Y T S D W Q S L G U C X A U B D
E E X C A O B C D E E U O K H G R K J Q A G P
N D N A I K O L U L A A N U K A Y T U R L P I
X N J M N E T R A I L U Q E T U K H B G F E I
S Y N A A G U L Q J H F C Y T S M A S R A H N
B C A O B C D E E O P Y M N Q S T U V O C O U
I G Y Z H M A W U K N G B Y I P U A S H H U I
L R B A R A A Q F J K X Y Q V N R T Y I F R Z
R I B V J K M P Y Z S I F U X C A B Q N P A H
C H G N Z X U Z J O N A T H A N Z W R A M J B
T P K L L K G R Z E R K L N H E Z S S W B E Z
K X L G U U V G W J J S Q K H J Z T I A T H J
Q R H O Q G N N Q Z R D U O J Y Y B A K D N O
A A E C A W U S W O C U Z M Y K U X T E A L V
G R T N T Y M R P B L F D T A C S F L L K U D
B R A L W W N X A E F Q G H K B H X C U P J H
Z V A N K D H R M O H Y O A V Z A D S P Z R S
I K N B U A O O Z D S M T Y H L N Z L D W M E
P E S J L F R D S T A X X C M A K H D E G V M
D C L E O Z W H I K O O A U I J U R X I T F O
C U G B I O J E R A I D A F Z I Y U S H L J S
B N H C C G T B B T N T I N T H C X K D D N Y
A Y O O N J A E Z Z O K J V K G D A F C V R D
The word is BEN
```

Word found at row 0 and column 0 in the south direction.

Another example

```
The word search puzzle is
S Q I F J Q K Y T S D W Q S L G U C X A U B D
W A X C A O M I G U E L O K H G R K J Q A G P
Z D L A I K O L U Q S F D V N S A L O H C I N
X N J M N E T R A I L U Q E T U K H B G F E I
S A H I L G U L Q J H F C Y T S M A S H A H W
B C A O B I D E E O P Y M N Q S T U V Z C O R
I G Y Z H M S W U K N G B Y I P U A S R H U I
L R F K Q Y U A F J K X Y Q V N R T Y O F R Z
R I B V J K M P Y Z S I F U X C A B Q W P A H
C H G N Z X U Z J O N A T H A N Z W R A M J B
T P K L L K G R Z K R K L N H E Z S S N B E S
K X L G U U V G A J J S Q K H J Z T I V T H H
Q R B R I A N F Q Z R D U O J Y Y B A W D N L
A A N C A W U S W O C U Z M Y K U X T N A L O
G R G K T I M R P B K F D T A C S F L L K U K
B R U L W W N X A F F Q G H K M H X C U P J J
Z B T N K D H K T O H Y O A V A A D S P Z R W
J K Z B U T O E Z D S M T Y H I N Z L D W M A
P E S J E V E L Y N A X X C M R K H D E G V I
D C L I O Z W L I K O O A U I A U R X I T F O
C U N B I O J Y B E T H E L Z M Y U S H L J T
B N H C C G T B B T N T I N T H C X K D D N Y
A M R O S E D V Z Z O K J V K G D A F C V R D
The word is TORONTO
Word not found.
```

Fun fact: You may find your name in these word search puzzles!

Part 2 — Sudoko

The famous Sudoko game is played on a grid of 9×9. Each empty square in this grid has to be filled with a number from $1 - 9$ without repeating any numbers within the row and column of the square. We will implement a program in `C` to help us solve a simple 4×4 Sudoko puzzle. We will investigate a 4×4 Sudoko puzzle, where *each* row or column is missing 2 numbers *only*. To fill in a square, if you scan the row (or column) of the square, you will eliminate two out of 4 numbers, and if you then scan the column (or row) of the square, you will eliminate one more number. Thus, you will be left with one missing number in the row and column of the empty square. Thus, you need to fill in the square with this missing number. We are NOT studying the case where scanning the row and column of a square eliminates only 2 numbers.

For example, in the following grid, row 0 and column 0 is empty. If we scan row 0, we can eliminate 3 and 4 since they are already there in the row. And if we scan the column, we can additionally eliminate 1. Hence, we can fill the square with 2.

```
0 3 4 0
4 0 0 2
1 0 0 3
0 2 1 0
```

After solving the Sudoko puzzle above, it will look like,

```
2 3 4 1
4 1 3 2
1 4 2 3
3 2 1 4
```

You are required to implement a function that fills in empty squares in the Sudoko grid. The function prototype is:

```
void fillSudoko(const int Size, int sudoko[Size][Size]);
```

`Size` is a named constant `const int Size = 4;` and `sudoko` is the 4×4 sudoko grid. You are encouraged to use other functions. For example, have a function that checks the row of a square and another to check the column of a square. Although you will only submit your functions, you are required to implement the whole program on VS Code to test it before submitting.

Hint: When you visit an empty square, you can have a "table" that holds which numbers are found in the row and column of the square. This will help you know which number is missing.

Marking

This lab will be marked out of 10, with 5 marks allocated to each part. Although your code will not be marked for style, your code should be readable so that your TA can help you during your lab session. Here are the basic requirements for your code to be readable:

- Use meaningful identifiers for the variables in your C function. Use either snake case or camel case, but be consistent throughout your entire C program.
- Use named constants when needed, rather than literal constants.
- Use consistent indentation throughout your entire C program.
- Use comments at the beginning of your program before the `main()` function. You may also add comments throughout your program as necessary, but they are not strictly enforced.
- Define and call functions as required in this laboratory.

The automarker may use multiple test cases for each part, and if this is the case, the marks for this part will be split among the test cases. The test cases used for marking may or may not be the same as the test cases that are made available to you. The deadline of **(11:59 p.m. on Saturday, March 12, 2022)** is strictly enforced, so avoid last minute submissions.

Stay Safe and Good Luck!