NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING

Practical Examination 2 (PE2) for Semester 2, AY2015/6
CS1010 Programming Methodology

30 March 2016 Time Allowed: 2 hours

INSTRUCTION TO CANDIDATES

1. You are only allowed to read this cover page and the last page. Do **not** read the questions until you are told to do so.
2. This paper consists of **2** exercises on **6** pages. Each exercise constitutes 50%.
3. This is an open-book exam. You may bring in any printed material, but **not** electronic devices, including but not limited to laptop, thumb-drive, electronic dictionary and calculator. You are to switch off/silence your mobile phone and keep it out of view.
4. You may turn to the last page (page 6) to read some advice.
5. You will be logged into a special Windows account at the beginning of the PE. Do not log off until the end of the PE. Do not use your own NUSNET account.
6. A plab account slip will be issued to you at the beginning of the PE. Bring your matriculation card for identification when you collect it. Please leave your matriculation card on the desk in front of you throughout the PE.
7. You are to write your program in the given **plab account**. The host name is **plab4** (not sunfire!). No activity should be done outside this plab account.
8. You do **not** need to submit your programs to CodeCrunch. We will retrieve your programs and submit them to CodeCrunch after the PE.
9. Skeleton programs and some test data files are already residing in your plab account. Please leave the programs in the home directory, and use the same program names as specified in the paper. Do **not** create subdirectory to put your programs there or we will **not** be able to find them!
10. **Only your source codes (.c programs)** from your plab account will be collected after the PE. Hence, how you name your executable files is not important.
11. Please read carefully and follow all instructions in the question. If in doubt, please ask. Raise your hand and an invigilator will attend to you.
12. Any form of communication with other students or the use of unauthorised materials is considered cheating and you are liable to disciplinary action.
13. Please save your programs regularly during the PE.
14. When you are told to stop, please do so **immediately**, or you will be penalised.
15. At the end of the PE, please **log out from your plab account**.
16. Please check and take your belongings (especially matriculation card) before you leave.
17. We will make arrangement for you to retrieve your programs after we have finished grading. Grading may take a week or more.

ALL THE BEST!

Exercise 1: Minefield [50 marks]

Problem Statement

During the World War II period, anti-tank mines were used to damage or destroy vehicles including tanks and armored fighting vehicles. As a counter-move, minesweepers were trained to deal with these mines.

One (fictional) way of doing so is to first survey the areas in a minefield to classify the mines by power. A mine of power ***k***, where ***k*** is a positive integer, deals damage to the area itself, as well as ***k***-1 area(s) in four directions (i.e., north, south, east and west). For simplicity's sake, damage dealt diagonally or to areas outside the minefield is ignored.

For example, as shown in Fig. 1, a mine of power 1 only deals damage to the area itself (without affecting the neighbouring areas). In contrast, a mine of power 2 deals damage to the area itself and 1 neighbouring area in four directions.

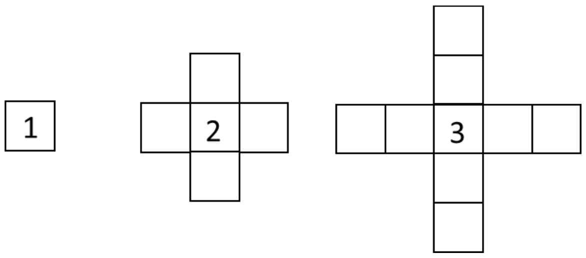

Figure 1. Areas of effect of mines of power 1, 2, and 3.

After mapping out the mines in a zone, the minesweepers assess the **aftermath** (i.e., the number of mines each area will be hit by if all the mines are detonated). They also compute the **maximum damage** sustained and the **number of safe areas**. Based on this information, they then decide the best strategy to deal with the mines.

For example, the minefield as shown below in Fig. 2a contains 4 mines: The two mines at [1][2] and [3][2] are of power 1, while the two at [2][1] and [2][3] are of power 2.

	[0]	[1]	[2]	[3]	[4]
minefield[0]	0	0	0	0	0
minefield[1]	0	0	1	0	0
minefield[2]	0	2	0	2	0
minefield[3]	0	0	1	0	0
minefield[4]	0	0	0	0	0

Figure 2a. A minefield with 4 mines.

	[0]	[1]	[2]	[3]	[4]
aftermath[0]	0	0	0	0	0
aftermath[1]	0	1	1	1	0
aftermath[2]	1	1	2	1	1
aftermath[3]	0	1	1	1	0
aftermath[4]	0	0	0	0	0

Figure 2b.The aftermath after all mines are detonated.

The areas affected by these mines are as shown in the table below, while the aftermath after all the mines are detonated is as shown above in Fig. 2b.

Location	Power	Area(s) affected
[1][2]	1	[1][2]
[2][1]	2	[2][1], as well as [1][1], [2][0], [2][2] and [3][1]
[2][3]	2	[2][3], as well as [1][3], [2][2], [2][4] and [3][3]
[3][2]	1	[3][2]

In the aftermath, the maximum damage sustained is 2 (i.e., [2][2] is hit by the two mines at [2][1] and [2][3]), while the number of safe areas is 14 since there are 14 areas which do not sustain any damage from any mine (e.g., [0][0]).

In this exercise, you are to write a program to 1) simulate the aftermath of the detonation of all the mines in a given minefield, and 2) compute the maximum damage sustained and the number of safe areas.

Your program should read in the size of the minefield ***n*** (1<=n<=10), as well as ***n****n*** numbers representing the locations and the powers of the mines. The numbers in this input are either 0 (no mine) or a positive integer ***k*** (a mine of power ***k***).

Your program should print the aftermath, the maximum damage sustained, and the number of safe areas as the output.

Write on the skeleton file **minefield.c** given to you. You must include the following function in your program:

- **`detonate()`**

  This function takes in the minefield, which is a 2D **int** array, as well as the size of the minefield, which is an integer. It returns the computed aftermath, as well as the maximum damage sustained and the number of safe areas.

You are to determine the return type and parameters for this function. You may define additional functions as needed. Check sample runs for input and output format.

Sample Runs

Three sample runs are shown below with user input highlighted in **bold**.

```
Enter size:
3
Enter minefield:
0 0 0
0 1 0
0 0 0
Aftermath:
0 0 0
0 1 0
0 0 0
Maximum damage: 1
Number of safe areas: 8
```

```
Enter size:
5
Enter minefield:
0 0 0 0 0
0 0 0 0 0
0 0 3 0 0
0 0 0 0 0
0 0 0 0 0
Aftermath:
0 0 1 0 0
0 0 1 0 0
1 1 1 1 1
0 0 1 0 0
0 0 1 0 0
Maximum damage: 1
Number of safe areas: 16
```

```
Enter size:
5
Enter minefield:
0 0 0 0 0
0 0 1 0 0
0 2 0 2 0
0 0 1 0 0
0 0 0 0 0
Aftermath:
0 0 0 0 0
0 1 1 1 0
1 1 2 1 1
0 1 1 1 0
0 0 0 0 0
Maximum damage: 2
Number of safe areas: 14
```

Exercise 2: NATO Alphabet [50 marks]

Problem Statement

The NATO alphabet is a set of code words assigned to English letters (as shown in Fig. 3) so that letter combinations can be pronounced and understood clearly.

A	B	C	D	E	F	G
ALFA	BRAVO	CHARLIE	DELTA	ECHO	FOXTROT	GOLF
H	I	J	K	L	M	N
HOTEL	INDIA	JULIETT	KILO	LIMA	MIKE	NOVEMBER
O	P	Q	R	S	T	U
OSCAR	PAPA	QUEBEC	ROMEO	SIERRA	TANGO	UNIFORM
V	W	X	Y	Z		
VICTOR	WHISKEY	XRAY	YANKEE	ZULU		

Figure 3. Letters and the assigned code words.

For example, to spell out the letter combination "TAN", instead of saying "T"-"A"-"N", we say "TANGO"-"ALFA"-"NOVEMBER". As another example, we spell out "AARON" as "ALFA"-"ALFA"-"ROMEO"-"OSCAR"-"NOVEMBER" instead of the individual letters.

In this exercise, you are given a string which is (possibly) a name spelt out in NATO alphabet. Your program should decrypt this string to find out what the name is.

Write on the skeleton file **nato.c** given to you. You must include the following function in your program:

- **`int decrypt(char name[])`**
  This function takes in a string **`name`**. It returns 0 if **`name`** cannot be decrypted using NATO alphabet. Otherwise, it returns 1 and the decrypted name in **`name`**.

You may define additional functions as needed. You may assume that the input is at most 100 characters long and contains only uppercase letters. Check sample runs for input and output format. Read the skeleton code for hints.

Sample Runs

Three sample runs are shown below with user input highlighted in **bold**.

```
Enter name in NATO alphabet: JULIETT
Decrypted name: J
```

```
Enter name in NATO alphabet: PANDAPO
The given name cannot be decrypted.
```

```
Enter name in NATO alphabet: ALFAALFAROMEOOSCARNOVEMBER
Decrypted name: AARON
```

CS1010 AY2015/6 Semester 2
Practical Exam 2 (PE2)

Advice – Please read!

- You are advised to spend the first 10 minutes for each exercise thinking and designing your algorithm, instead of writing the programs right away.
- If you write a function, you must have a function prototype, and you must put the function definition after the main() function.
- You may write additional function(s) not mentioned in the question, if you think it is necessary.
- Any variable you use must be declared in some function. You are **not** allowed to use global variables (variables that are declared outside all the functions).
- You may assume that all inputs are valid, that is, you do not need to perform input validity check.
- Manage your time well! Do not spend excessive time on any exercise.
- Be careful in naming your executable code. Do **not** overwrite your source code with your executable code, especially if you are using the –o option in gcc!
- The rough marking scheme for both exercises is given below.

Rough Marking Scheme For Each Exercise [50 marks]

1. Style: 10 marks
   - Are name, matriculation number, plab-id, DG and description filled at the top of the program?
   - Is there a description written at the top of every function (apart from the main() function)?
   - Are there proper indentation and naming of variables?
   - Are there appropriate comments wherever necessary?
2. Design: 10 marks
   - Are there correct definition and use of functions?
   - Are function prototypes present?
   - Is the right construct used?
   - Is algorithm not unnecessarily complicated?
3. Correctness: 30 marks
4. Deductions (not restricted to the following):
   - Program cannot be compiled: Deduct 5 marks
   - Compiler issues warning with –Wall: Deduct 5 marks.
   - Use of global variables: Deduct 10 marks

--- END OF PAPER ---