NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING

Practical Examination 2 (PE2) for Semester 1, AY2015/6
CS1010 Programming Methodology

31 October 2015 Time Allowed: 2 hours

INSTRUCTION TO CANDIDATES

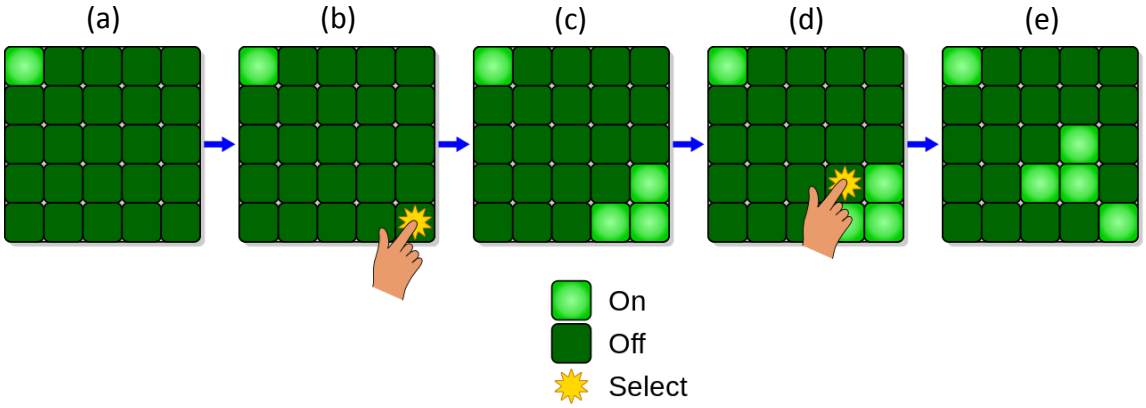
1. You are only allowed to read this cover page and the last page. Do **not** read the questions until you are told to do so.
2. This paper consists of **2** exercises on 8 pages. Each exercise constitutes 50%.
3. This is an open-book exam. You may bring in any printed material, but **not** electronic 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 8) 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 the 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: Lights Out! [50 marks]

Problem Statement

Lights Out is an electronic game released by Tiger Electronics in 1995. The game consists of a 5x5 grid of panels. When the game starts, some of the panels are lighted. Pressing a panel will toggle it and the adjacent panels. The goal of the game is to switch all the panels off by pressing some of them in sequence.



In this exercise, you are to write a program to simulate a sequence of panel presses on a given grid and determine the final state of the grid.

For example, if the initial state of the grid is as shown in (a) in the sample diagrams above (i.e., only the panel at the upper-left corner is lighted), after pressing two panels as shown in (b) and (d), the final state of the grid is as shown in (e) (i.e., a total of 5 panels are lighted).

Your program should read in a 5x5 grid of numbers representing the initial state of the grid. The numbers in this input are either 0 (not lighted) or 1 (lighted). For example, the grid shown in (a) is represented as follows:

Panels 0~4	1	0	0	0	0
Panels 5~9	0	0	0	0	0
Panels 10~14	0	0	0	0	0
Panels 15~19	0	0	0	0	0
Panels 20~24	0	0	0	0	0

Your program should also read in the length of the sequence of panels to be pressed, as well as the sequence itself.

The length of the sequence is an integer between 1 and 30 (i.e., there is at least 1 and at most 30 panels in the sequence).

The actual sequence is a sequence of integers between 0 and 24, each corresponding to a particular panel to be pressed. For example, 0~4 correspond to the five panels (from left to right) in the top row of the grid, while 20~24 correspond to the five panels in the bottom row. In the sample diagrams, the panel pressed in (b) is 24, whereas the one pressed in (d) is 18.

Your program should follow the sequence of presses to determine the final state of the grid after all the presses. For example, the final state for the example shown in the sample diagrams is as follows:

1	0	0	0	0
0	0	0	0	0
0	0	0	1	0
0	0	1	1	0
0	0	0	0	1

Write on the skeleton file **panel.c** given to you. You must include the following two functions in your program:

- **`press()`**

  which takes in the grid, the length of the sequence, and the actual sequence. It follows the sequence of presses and updates the grid accordingly.

- **`allOff()`**

  which takes in the grid, and returns 1 if all the panels are not lighted. Otherwise, it returns 0, as well as the number of panels which are lighted.

You are to determine the appropriate return types and parameters for these two functions. You may define additional functions as needed. Check sample runs for input and output format.

Sample Runs

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

Set #1:

```
Enter grid:
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
Enter length of sequence: 1
Enter sequence of panels: 12
Number of lighted panels: 5
0 0 0 0 0
0 0 1 0 0
0 1 1 1 0
0 0 1 0 0
0 0 0 0 0
```

Set #2:

```
Enter grid:
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
Enter length of sequence: 1
Enter sequence of panels: 12
Number of lighted panels: 20
1 1 1 1 1
1 1 0 1 1
1 0 0 0 1
1 1 0 1 1
1 1 1 1 1
```

Set #3:

```
Enter grid:
0 0 0 0 0
0 0 1 0 0
0 1 1 1 0
0 0 1 0 0
0 0 0 0 0
Enter length of sequence: 1
Enter sequence of panels: 12
All panels are off.
```

Set #4:

```
Enter grid:
1 1 0 0 0
1 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
Enter length of sequence: 1
Enter sequence of panels: 0
All panels are off.
```

Set #5:

```
Enter grid:
1 0 0 0 0
0 1 1 0 0
0 1 0 0 0
0 0 0 0 0
0 0 0 0 0
Enter length of sequence: 2
Enter sequence of panels: 6 0
All panels are off.
```

Exercise 2: Email Address [50 marks]

Problem Statement

A company plans to develop an application for NUS students to buy and sell their second-hand textbooks. To make sure that the users are NUS students, the application should be able to check whether an email addressed used for registration is a valid NUS email address. In addition, it should also be able to check whether the email address used is a friendly email address, so that it can remind the users to update the friendly email addresses if there are any changes.

You have been hired by this company to write a part of this application. Your program should read in an input string and check whether the input is a valid NUS email address. If so, it should also check whether the input is a friendly email address.

After searching for relevant information online, you have found out that an email address is a **valid NUS email address** if all of the following criteria are met:

- The email address should consist of two parts: a local part and a domain part. These two parts are separated by the symbol '@'.
- The local part of the email address should be 3 to 21 characters long (both inclusive). It must start with a letter but the rest of it can be letters, digits or periods ('.').
- The domain part should be "u.nus.edu".

For example, "benedict.90@u.nus.edu" is a valid NUS email address.

In addition, you have also found out that a **friendly email address** is a valid NUS email address whose local part is **not** an 'a' or an 'e' followed by 7 digits. For example, "benedict.90@u.nus.edu" is a friendly email address but "a1234567@u.nus.edu" is not.

You may assume that the input is at most 50 characters long and does not contain any uppercase letters or white spaces.

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

- **`int checkEmail(char email[])`**
  which takes in an input string **`email`**. It returns 0 if **`email`** is not a valid NUS email address, 1 if **`email`** is valid but not friendly, or 2 if **`email`** is both valid and friendly.

You may define additional functions as needed. Check sample runs for input and output format.

Sample Runs

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

Set #1: // Meets all criteria

```
Enter email address: benedict.90@u.nus.edu
This email address is valid and friendly.
```

Set #2: // Not friendly because the local part is 'a' followed by 7 digits

```
Enter email address: a1234567@u.nus.edu
This email address is valid but not friendly.
```

Set #3: // Too short

```
Enter email address: a@u.nus.edu
This email address is not valid.
```

Set #4: // Contains illegal character '+'

```
Enter email address: a+b@u.nus.edu
This email address is not valid.
```

Set #5: // Wrong domain

```
Enter email address: bobby@nus.edu.sg
This email address is not valid.
```

CS1010 AY2015/6 Semester 1
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 ---