NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING

Practice Paper 2 for Practical Assessment
CS1010J — Programming Methodology

Time Allowed: 1.5 hours

INSTRUCTIONS TO CANDIDATES

1. This assessment paper consists of 2 exercises on 4 pages.
2. This is an **OPEN BOOK** assessment. You may bring in any printed material and a calculator, but **not** other electronic devices such as laptops, iPads or thumb-drives.
3. In line with the university rules, any form of communication with other students, or the use of unauthorised materials is considered cheating and you are liable to the disciplinary action.
4. The first 10 minutes of the assessment is reserved for you to read questions and design algorithms. You are not allowed to type your programs in this period of time.
5. You are to use **DrJava** or **VS Code** to write your programs. No other IDEs are allowed.
6. You may assume that all input data are valid; hence no input validation is needed.
7. Once finish, you may submit your programs to **CodeCrunch** for automatic grading.
8. You are **NOT** allowed to use any Java language syntax not covered in weeks 1-9 (e.g., string functions or recursion).

ALL THE BEST!

Exercise 1: Arrange Integers [40 marks]

Write a program **ArrangeIntegers.java** to complete the following two tasks:

1. check if a given integer array contains duplicate integer (i.e. an integer that appears more than once).
2. rearrange the given integer array as follows.
    a. All negative integers (if any) to be moved to the front of the array.
    b. All positive integers (if any) to be moved to the back of the array.
    c. Zeroes (if any) to be placed between negative and positive integers.
    d. Relative order of negative and positive integers must be preserved. For example, if a negative integer x initially appears before another negative integer y, after re-organizing x should still be in front of y.

Four examples are given below to help you better understand the requirements:

- Array {1} contains no duplicate integer; it will be rearranged as {1}.
- Array {1, -2, 3, -4} contains no duplicate integer; it will be rearranged as {-2, -4, 1, 3}.
- Array {-1, 9, -4, 0} contains no duplicate integer; it will be rearranged as {-1, -4, 0, 9}.
- Array {0, -5, 7, 0, 3, -5} contains two duplicate integers (-5 and 0); it will be rearranged as {-5, -5, 0, 0, 7, 3}.

In the given skeleton program, you are supposed to complete the following two methods:

- **`boolean containsDuplicate(int[] arr)`** that returns true if **`arr`** contains at least one duplicate integer, or false otherwise.
- **`void rearrange(int[] arr)`** that rearranges **`arr`** as described in Point 2 above. You may define additional arrays in this method as necessary.

You must **NOT** change the given **`main()`** method or the method headers of all other methods. You may write additional methods as necessary.

Two sample runs are shown below with the user's input shown in **bold**.

Sample run #1

```
Enter the number of integers: 4
Enter 4 integers: 1 -2 3 -4
Array doesn't contain duplicate integer
Rearranged array: [-2, -4, 1, 3]
```

Sample run #2

```
Enter the number of integers: 6
Enter 6 integers: 0 -5 7 0 3 -5
Array contains duplicate integer
Rearranged array: [-5, -5, 0, 0, 7, 3]
```

Exercise 2: Colour Bomb [60 marks]

Candy Crush is a popular board game that attracts much attention. The game is played by swapping adjacent (i.e. left, right, above or below) candies to create sets of consecutive matching candies. A special candy called "Colour Bomb" will be created if, after swapping, **five or more** candies of the same type are found to lie in the same row or column.

The following example shows a 6*6 square board filled with six types of candies (indexed from 1 to 6). A Colour Bomb can be created after swapping the two highlighted candies, because five consecutive 3s will be formed in the second last row after swapping.

1	2	1	1	4	5
4	1	2	3	3	3
3	1	3	3	1	2
1	1	2	4	3	2
4	3	3	3	2	3
6	1	1	1	4	1

The following example shows another 6*6 square board that can form a Colour Bomb after swapping the two highlighted candies.

1	2	1	6	4	5
1	1	2	3	3	3
1	6	3	3	1	2
1	1	2	4	3	2
4	1	2	3	2	3
1	1	1	1	4	1

Write a program to check if a Colour Bomb can be created after swapping any two adjacent candies. In the skeleton program **ColourBomb.java**, complete the following four methods.

- **`int[][] readBoard()`** that creates a square game board (of at least 5 rows and 5 columns) from user input and returns it. The board is filled with integers in the range 1 - 6, representing 6 types of candies.

- **`void swap(int[][] board, int r1, int c1, int r2, int c2)`** that swaps the candy at row **`r1`**, column **`c1`** with the candy at row **`r2`**, column **`c2`**.
- **`boolean hasStraightFive(int[][] board)`** that returns true if **`board`** contains at least one set of five candies of the same type in a row or column. It returns false otherwise.
- **`boolean canFormColourBomb(int[][] board)`** that checks if a Colour Bomb can be created after swapping a candy with one of its adjacent (left, right, above or below) candies. This method returns true if so, or false otherwise. You may assume that no Colour Bomb can be formed on the **`board`** prior to the swapping.

Take note of the following coding restrictions:

1. You must **not** change the method headers given in the skeleton program.
2. You are **not** allowed to change the **`main()`** method as well.
3. The game **`board`** is created in the **`readBoard()`** method and passed to other methods as a parameter. You may modify it as appropriate.

Two sample runs of the program are given below with the user's input shown in **bold**. For your convenience of testing, test inputs are also appended to the back of the skeleton program.

Sample run #1

```
Enter the size of the board: 6
Enter the board:
1  2  1  1  4  5
4  1  2  3  3  3
3  1  3  3  1  2
1  1  2  4  3  2
4  3  3  3  2  3
6  1  1  1  4  1
At least one Colour Bomb can be created after swapping
```

Sample run #2

```
Enter the size of the board: 7
Enter the board:
1 5 2 3 1 4 4
4 1 2 2 2 2 6
3 1 2 3 1 2 5
1 1 2 4 3 2 2
3 3 3 4 2 3 5
2 1 1 3 4 3 6
1 1 1 5 1 1 1
No Colour Bomb can be created after swapping
```

=== END OF PAPER ===