CSE244, Lab07, Scheduling a Hockey Tournament with a 2-dimensional array

1. General Aims

The purpose of this assignment is to use 2-dimaensional arrays to solve problems.

2. Deliverables and Deadline

This lab contains one task (see 3). One email attached with your compressed Android project file (compressed from your Android project) should be sent to liang-y@rmc.ca no later than Nov. 16, 2016. In the subject area of your email, key in the course code "CSE244 LAB07" in order to file your email in my designed folder easily.

3. Task

[**Task 1**] Write a program, which produces a schedule for a hockey tournament according to the following rules:

a. There will be n teams playing
b. The tournament will last n days
c. A team may only play one game per day
d. Each team must play every other team once

Your program must work for values of n from 3 up to 20 (inclusively) and must be written following the design shown in the algorithm as follows:

(1). Read in an integer `n` (`3≤n≤20`), and create a 2-dimensional array:

```
int[][] match = new int[n+1][n+1];
```

Here `i` represents Team `i` and `j` for Day `j`. The reason to have `(n+1)×(n+1)` elements is that we will not use those elements where `i=0` or `j=0` because Team 0 and Day 0 do not make sense in this exercise. The value of `match[i][j]` stores the team number against whom Team i will play on Day j. For example, `match[1][1] = 2` represents Team 1 on Day 1 will play against Team 2. A value 0 for an element of `match[i][j]` indicates Team i on Day j will not play against anyone, or Team i meets other team in previous calculation.

(2). Initialization: `match[i][j]` is set to zero for all `1≤i≤n` and `1≤j≤n`.
(3). The pattern of how to fill the 2-dimentional array match can be illustrated in the table below (assuming `n=7`):

Match / Day j / Team i	1	2	3	4	5	6	7
1	**2**	**3**	**4**	**5**	**6**	**7**	**0**
2	**0**	**0**	**3**	**4**	**5**	**6**	**7**
3	**7**	**0**	**0**	**0**	**4**	**5**	**6**
4	**6**	**7**	**0**	**0**	**0**	**0**	**5**
5	**0**	**6**	**7**	**0**	**0**	**0**	**0**
6	**0**	**0**	**0**	**7**	**0**	**0**	**0**
7	**0**	**0**	**0**	**0**	**0**	**0**	**0**

The red colour number refers to the day we start to calculate when Team `i` should play against Team `i+1`, one next day, play against team `i+2`, etc., until against Team `n`.

For Day 1, Team 1 plays against Team 2. Day 2, Team 1 against Team 3, etc. But for Day 7, Team 1 will have no game.

(4). For Team i ($2 \le i < n$):

- The day on which you begin the calculations for Team `i` against Team `i+1` is two days later than the day for Team `i-1`. For example, to calculate on which day Team 2 should play against Team 3, we could find that `j=1+2=3` while 1 is the day when we begin to calculate when Team 1 should play Team 2. Therefore, the value for `match[2][3]=3`.
- Team `i` plays against each team against which it has not already played. To calculate the entries for Team `i`, start with Team i + 1 and continue until you reach team `n`. Increase the day of the game by module (%) n for each team.

(5). Show the time schedule for each day, for example, if a user chooses n=7, he/she should have the following output on the screen:

```
Match Schedule for 7 Teams over 7 Days:
Day 1 matches:
        1-2
        3-7
        4-6
```

```
Day 2 matches:
        1-3
        4-7
        5-6
Day 3 matches:
        1-4
        2-3
        5-7
Day 4 matches:
        1-5
        2-4
        6-7
Day 5 matches:
        1-6
        2-5
        3-4
Day 6 matches:
        1-7
        2-6
        3-5
Day 7 matches:
        2-7
        3-6
        4-5
```

(6). For the app, you have to add proper GUI components for allowing a user to read n, display match schedule, and start the calculation.

Deliverables

Submit a screenshot of your interface inserted into a word file, `MainActivity.java` and `activity_main.xml` files to liang-y@rmc.ca before the deadline.

Deadline

Nov. 30, 2018.

Marking Scheme:

item	Mark contents	Mark allocated	Mark obtained

1	Proper Android GUI is created for reading and displaying.	2	
2	A method doing the scheduling is designed.	2	
3	Program work properly	2	
4	The number of teams n should be ≤ 20	2	
5	2D array is used	2	
	Total:	10	