Temasek Polytechnic
School of Informatics & IT
Diploma in Information Technology
Diploma in Mobile & Network Services
Diploma in Interactive Media Informatics
Diploma in Cyber & Digital Security
Diploma in Game & Entertainment Technology
Diploma in Financial Business Informatics
Diploma in Business Intelligence & Analytics
Problem Solving & Programming (CIT1C05)
AY 12/13 April Semester (Level 1)

Started **Week 10** *Due* **Week 16 in lab**

Assignment

This is an **individual** assignment. Your submission must not contain any plagiarized materials especially the codes.

Introduction / Aim

This assignment requires students to put in practice the various topics covered so far regarding programming.

1. Can You Open the Lock?

Write the **pseudo-code** for a program to simulate opening a combination lock. The combination lock consists of 3 numbers.

The program should generate three random numbers between 10 and 40, inclusive. These three numbers will be the first, second, and third number of the combination lock. Then, the program should generate three random numbers between 5 and 45, inclusive. These 3 numbers are the first, second and third numbers used to try and open the lock.

If all these three numbers tried are within plus or minus two of the combination lock numbers, then print "Lock opens!" Otherwise print "Unable to open lock. Try again!"

A sample run of the program is shown below:

```
Combo first number        : 10
Combo second number       : 20
Combo third number        : 40

Tried first number        : 12
Tried second number       : 18
Tried third number        : 42

Lock opens!
```

Translate the pseudo-codes into a program using a programming language. (*Hint*: To have the computer decide a random response, use the appropriate random number generation function).

2. What's Your Level?

Write the **pseudo-code** for a program to identify the merit level of a student at a university as in the Table 2a:

Assume the merit award for student activities range from 0 to 5, leadership merit points range from 0 to 10, cGPA ranges from 0 to 5 and attendance ranges from 0 to 100. All input for this question should be randomly generated and stored in appropriate variables (assume all variables used are of data type integer).

	cGPA(0-5)	Attendance in all Subjects (0-100%)	Merit Awards in Student Activities (0-5)	Leadership Merit Points (0-10)	Merit Level
1	>=3	100	>=2	>=7	High Merit
2	>=3	100	1	3 to 6	Medium Merit
3	>=3	100			Merit
4	2	100	Either at least 2 points in Merit Awards in Student Activities OR at least 1 point in Leadership Merit points		Participating Merit
5	Others				No Merit Award

Table 2a

Translate the pseudo-code into a program using a programming language.

3. How many?

Write the **pseudo-code** for a program, using a **for** loop, to generate 30 random numbers between 20 and 60, inclusive. Then, the program should display how many of those numbers are smaller than or equal to 35 and how many are larger than 45.

Translate the pseudo-code into a program using a programming language.

4. What to replace?

Write the **pseudo-code** for a program to declare and create an integer array of size 10. Then, the program should randomly, using a **while** loop, insert in numbers, in the range of 1 to 50, inclusive, into each slot of the array. The program should print the array values. Next, the program should randomly generate an integer, between 1 and 50, inclusive, to search for and also print it as output. Then, the program should replace all the numbers larger than the integer in the array with the value -1. If that integer does not appear in the array, then the program does not change any of the items in the array. At the end, the program should print the final array as output using a **while** loop. (Assume there is **no user input** for this question.)

Translate the pseudo-codes into a program using a programming language.

A sample run of the program is shown below:

Sample output #1:
```
Array:    43   14   8    12   1    23   15   4    2    14
Value:    20
Array:    -1   14   8    12   1    -1   15   4    2    14
```

Sample output #2:
```
Array:    3    4    8    2    10   21   15   4    2    14
Value:    7
Array:    3    4    -1   2    -1   -1   -1   4    2    -1
```

5. Which Has More Even Numbers?

Write a program to create two arrays of integers of size 10. Then, the program should generate and insert random numbers between 1 and 25, inclusive into both arrays. Next, the program should find out the number of even numbers in each array and print the array having more even numbers (i.e. if it is the first array, the program should print "First Array", otherwise print "Second Array"). In the case of a tie, the program should print "Tie". Your program must include, at least, the following methods:

- **generateRandomInput**, which will take as input one integer array and generate and store the random numbers in the array.
- **countEven**, which will take as input one integer array and return the number of even values in the array.
- **printResult**, which will take as input the number of even numbers of both arrays and print the final output.

6. Connect 4!

Write a program to simulate a simplified version of Connect 4. Connect 4 is a two player game played on a 6x6 game board, where the objective is to drop circular pieces into 6 column slots. The winner is decided when one of the two computer players is able to get 4 circular pieces together horizontally or vertically. This modified version of Connect 4 will be simulated using two computer players.

Assume there is **no user input** for this game. The 6x6 board will be simulated using a 6 x 6 2D array. At the start, the game must randomly decide which computer player (1 or 2) will go first. Then, the program should simulate the game play as follows:

- At each turn, the current state of the game board should be displayed
- At each turn, each computer player randomly drops a circular piece into one of the 6 column slots. The column to be dropped into must be randomly decided. Each column slot can only take a maximum of 6 pieces. Then, when a circular piece is dropped in, it will go into a position in that column, which has a x and y value (e.g. [2][2]). If that column is full, then the computer player skips the turn.
- After each circular piece is dropped, the program must check if that computer player has won or not. A win means that after that drop there is either a horizontal or vertical group of 4 circular pieces on the game board.
- If there is no winner, then at the end of the turn, the updated state of the 6x6 2D game board should be displayed.

Insert circular piece ⇩ ⇩ ⇩ ⇩ ⇩ ⇩

(x,y) →

(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(5,0)
(0,1)	(1,1)	(2,1)	(3,1)	(4,1)	(5,1)
(0,2)	(1,2)	(2,2)	(3,2)	(4,2)	(5,2)
(0,3)	(1,3)	(2,3)	(3,3)	(4,3)	(5,3)
(0,4)	(1,4)	(2,4)	(3,4)	(4,4)	(5,4)
(0,5)	(1,5)	(2,5)	(3,5)	(4,5)	(5,5)

Overall, each position (or slot) in the displayed 6x6 2D game board can contain any of the following:

- -1 → not occupied yet
- 1 → occupied by computer player 1's circular piece
- 2 → occupied by computer player 2's circular piece

The game ends when one of the following conditions is satisfied:

- Either computer player 1 or 2 has a horizontal or vertical group of 4 pieces on the game board. If this condition is satisfied the program should print that computer player as the winner.

- There are no more slots in the game board to drop pieces onto. Then, the program should print “No winner” as output.

Your program must, **at least**, include the following methods/functions with the correct input parameters and output type:

i. **initializeGameBoard**, which will take as input the player’s 2D array and assign blank slots containing -1 (-1 is assumed to be unoccupied yet).

ii. **dropPiece**, which will drop a circular piece randomly into a slot in one of the six columns.

Your program also needs to track and print the following:
- Number of pieces dropped into each column by computer player 1
- Number of pieces dropped into each column by computer player 2

Marks will be awarded for further making your **program modular**. You must carefully choose the methods.

Status Report

The status report must be typed-written using Microsoft Word with 1 inch margin all round, portrait, Times New Roman font, size 12, single line spacing.

Write a short summary of the final state of your assignment in a file called Status_<YourName>_<AdminNumber>.doc.
E.g. *Status_LimChuKang_0709115J.doc.*

For **each** program, state the following:
- Any assumptions made
- Any limitations in the programs
- Any improvements that can be made to the programs

Submission

Submit the following items**:**
1. **soft copy** stored in CD & **hard copy** of your program code
2. **hard copy** of the report (should be typed and printed out)

Put the above items in an A4 envelope with the subject name, your name, your admin number and your lab group neatly written on the envelope cover. The submission time is the **first 15 minutes of the lab session in week 16.** Assignment will be returned to you after marking.

Penalty for late assignment submission

late and < 1 day: 10% deduction from absolute mark for the assignment
late >=1 and <2 days: 20% deduction from absolute mark for the assignment
late >=2 days: No marks will be awarded

Note: The “day” definition includes **non-working days** (ie. Sat, Sun and public holidays).

General MC/LOA is NOT considered as valid reason for not submitting your assignment.

Warning

You must ensure that Assignment submitted by you is your **original** work. Any **suspected copy cases** will be handed over to School Disciplinary Committee. You may **fail the subject** for this semester as a result of plagiarism.

Marking Scheme for assignment submission

Total: 100 marks (30% of overall PRSP marks)

Report: (12 marks)

Program Implementation/Coding: (88 marks)

Programs	**Marks**
1. Can you open the lock?	10
2. What's your level?	10
3. How many?	10
4. What to replace?	10
5. Which has more even numbers?	15
6. Connect 4!	25
Programming Style (Documentation/Comments) Additional Features	8

Penalty

You will be penalised for the following:

- syntax, link and runtime errors (so test early and test often!)
- lack of comments and missing header comments
- poor programming styles (this includes indentation, variable naming conventions etc)
- poorly written/formatted status report
- failure to comply with submission instructions and standard file naming convention
- late submission.

Assignment - Post-Submission Lab Test (25%)

During week 17, a **lab test** will be conducted during the lab session. Your lab test will test you on the same topics you have used as part of your Assignment submission.

The Assignment submission together with the Post-Submission Lab Test will constitute 55% to your continuous assessment and is to be submitted in the first 15 minutes of your week 16 lab session.

**************END OF ASSIGNMENT****************