ACS-1903
Assignment 2
Due by Friday November 13 at 11:59 pm
- Submit your .java files via Nexus
- Include your name and student number in each file as a comment

Note: concepts that have not been covered in class cannot be used e.g. break or continue statements, arrays, ArrayLists, etc.

This assignment is worth 10 % of your final grade, and is out of 100 marks. Each question is evenly weighted (20 marks).

1. Refactor the `DaysInAMonth` program from Lab 6 so its main method becomes:

```
import java.util.Scanner;
/**
 * ACS-1903 Lab 6 Q9
 * @author (.....)
 */
public class DaysInAMonth {
    public static void main (String [] args) {
        int month = getMonthFromUser();
        int days  = getNumberOfDays(month);
        displayResults(month,days);
        System.out.println("program terminated normally");
    }
}
```

2. Write a program named `PasswordEncryption` that will get a 6-9 character password made up of only letters and digits. Pass the password into a method called `validPW` that returns `true` if the password is 6-9 characters and contains only letters and digits, `false` otherwise. To validate the password characters you can use only the `isLetter()` and `isDigit()` methods of the `Character` class.

   Encrypt a valid password character by character according the following formula:
   - Get the numeric value of the character. Not the ascii or Unicode value.

- Square the value
- Multiply it by a random value from 350-400, all characters in the password should be multiplied by the same random number.
- Mod by 94
- Add 33

Print the original password and the encrypted password.

Sample output 1:
```
Enter your new password.
BigTuna12

Original PW: BigTuna12
Encrypted PW: 'iEi-CaKk

end of program
```

Sample output 2:
```
Enter your new password.
BigTuna12

Original PW: BigTuna12
Encrypted PW: ^ku<="9T1

end of program
```

3. Write a Java program named `RandomPowers` with 2 methods `getRandomNumber` and `getPower`.
   The header for `getRandomNumber` is
   `public static int getRandomNumber(int m, int n)`
   `getRandomNumber` returns a random number between n and m.

   The header for `getPower` is
   `public static int getPower(int r1, int r2)`
   `getPower` returns $r1^{r2}$ if r1>r2
   $r2^{r1}$ if r2>r1.
   Your program must
   a. use getRandomNumber twice to get 2 random numbers between m and n
   b. use getPower passing in the numbers from step a)

c. Output the result to the terminal window

d. After displaying the results from the above steps a) and b), ask the user if they want to repeat the process. The process is to be repeated over and over until the user enters a word beginning with N or n

Some restrictions:

i. In your program declare n and m as `final`. For example:
   `final int n = 9;`
   A final variable is actually a constant; once declared as above, its value cannot be changed later in the program. Use values of 4 and 9 for m and n.

ii. The loop in d) must be implemented using do-while

iii. getPower must not use if-else or switch statements to determine the largest and smallest random numbers - it must use max or min from either Math or Integer

iv. getPower must use `Math.pow()` to raise a number to a power.

v. The stopping condition of the loop cannot use logical operators && or ||.

You may include other methods as required.

```
Sample output:
7^5 = 16807
Calculate another power? (yes/no)
Yes
5^4 = 625
Calculate another power? (yes/no)
y
6^6 = 46656
Calculate another power? (yes/no)
No
Program Over
```

4. A valid course ID at the University of Winnipeg contains 3 parts:
   a) Department code: a series of letters, can range in length from 2 to 4 characters
   b) Course number: 4 characters, all numeric
   c) Section number: 3 characters, all numeric

   All parts are separated by a hyphen, e.g. ACS-1903-001, ENGL-3113-050

Write a program named `CourseIDValidation` that reads a list of course IDs from a file named `courses.txt` and determines validity. If an ID is valid, display that it is valid, otherwise display that it is invalid. For valid courses listed in the following chart, also display the associated faculty:

Dept code	Faculty
ACS	Science
CHEM	Science
MATH	Science
ENGL	Arts
IS	Arts
SOC	Arts

Sample data that may be in `courses.txt`

```
ACS-1903-001
ACS-190-050
BUS-1201-001
ENGL-2703-003
ENGLL-2703-002
CHEM1200-002
IS-1002-003
MATH-3902-00A
math-2301-050
KIN-2200-002
SOC_1201_002
SOC-1201-0020
```

Sample output:

```
ACS-1903-001 is a valid course number (Science).
ACS-190-050 is invalid.
BUS-1201-001 is a valid course number.
ENGL-2703-003 is a valid course number (Arts).
ENGLL-2703-002 is invalid.
CHEM1200-002 is invalid.
IS-1002-003 is a valid course number (Arts).
MATH-3902-00A is invalid.
math-2301-050 is a valid course number (Science).
KIN-2200-002 is a valid course number.
SOC_1201_002 is an invalid course number.
SOC-1201-0020 is an invalid course number.
```

Include 2 methods:

- `validate` - accepts a string and returns the boolean result of the validation test.
    - e.g. `boolean b = validate(courseID)` where courseID is the string to validate
- `displayResult` - a void method that accepts the original input string and boolean result of validate(); displays the result of an entry.
    - e.g. `displayResult(courseID, b)`

Some restrictions:

i. Must use a loop for validation.
ii. Must include the required methods
iii. Sample data is provided, but your program must work with any data

5. Write a program named `DiceGameSim` that simulates a simple dice game between 2 players. In this game, each player rolls a single die against the other player for 10 total rolls per round. The player with the higher roll wins a point (no point is rewarded for tie rolls). The score for a round is tallied over 10 rolls, and each round results in a winner or tie. The first player to win 3 rounds wins the game.

    Simulate a game between 2 players by randomly generating each dice roll. Display the results in the following format (each row represents a round of 10 rolls):

```
Rolls (player1-player2):
1    2    3    4    5    6    7    8    9    10   round
winner/score
1-5  3-3  1-1  6-5  1-6  1-2  1-3  3-4  4-1  5-1  player2 3-5
3-2  3-1  3-6  2-2  2-6  3-3  1-6  6-3  3-3  4-2  player1 4-3
1-3  3-3  6-5  4-3  3-6  5-5  1-6  6-3  6-1  2-6  --tie-- 4-4
1-4  1-2  4-1  3-2  3-6  5-4  2-3  5-5  2-5  6-4  player2 4-5
3-1  6-6  4-5  3-2  4-4  3-4  2-4  4-2  6-2  4-1  player1 5-3
1-2  3-1  1-4  6-5  1-1  2-6  3-4  3-3  4-4  2-4  player2 2-5

Player2 wins the game!
```

Some restrictions:

i. Must use Math or Integer utility method to find the winning roll
ii. Must include at least 1 method
iii. Must use the appropriate control structures

Submit your files (`DaysInAMonth.java, PasswordEncryption.java, RandomPowers.java, CourseIDValidation.java and DiceGameSim.java`) **via Nexus**.