CSE220 Assignment 2

Deadline: 17th May, 2021

Assignment Guidelines:

1. **Submission Instructions:**

    - **The assignment can be hand-written or typed.**
    - **Submission Format: PDF (one single file).**
    - **File naming convention: StudentID_StudentName_AssignmentNumber (Example: 18102541_BillGates_Assignment2)**
    - **Write down only the methods that are required; testers are not required.**

2. **Late Policy:** **Late assignments receive no grade, so please start early.**

3. **Policy on Academic Honesty:**

    Whenever you submit a piece of academic work and sign your name to it, you are verifying that this work is the result of your own intellectual efforts; it is Plagiarism to submit work solely under your own name in which:

    - **You collaborated with another student in solving the homework problems;**
    - **You copied the solution from a book, web site, or the work of another student; or**
    - **You got significant help in solving the problem, but do not acknowledge this in your solution.**

    In this course, you are not allowed to submit any joint work, although in other courses and in other contexts, you may be permitted to do this. In general, you must always provide proper attribution of authorship by naming all persons, books, or resources that provided intellectual content towards the final result. In some cases, you will need to describe the extent of the contribution, particularly when literally copying the words or artistic artifacts of another. To fail to provide proper attribution is plagiarism. Plagiarism demeans the seriousness of what we do in class, and does not allow you as a student to obtain a fair grade for the results you worked hard for.

Assignment Tasks:

1. [**Very Easy**] **[5 Marks]**

   Given **base** and **n** that are both 1 or more, **compute recursively** (no loops) the value of base to the n power, so powerN(3, 2) is 9 (3 squared).

   powerN(3, 1) → 3
   powerN(3, 2) → 9
   powerN(3, 3) → 27

2. [**Easy**] **[5 Marks]**

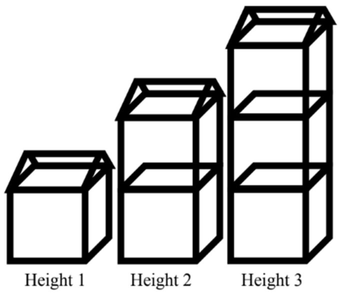
   Trace the following recursion code. Draw and Write down the outputs.
   Please, show the recursive flow in details to get the full mark.

```
public class Trace{
  public static void main(String [] args){
    System.out.println( "Finally " + hMB(5));
  }

  public static int hMB(int h){
    if(h==0){
      System.out.println("Stop: " + h);
      return 0;
    }
    else if(h==1){
      System.out.println("Stop: " + h);
      return h;
    }
    else{
      System.out.println("Continue: " + h);
      return h + hMB(h-1);
    }
  }
}
```

OR

```
class Trace:
    def hMB(self,h):
        if (h==0):
            print("Stop: ",h)
            return 0
        elif(h==1):
            print("Stop: ",h)
            return h
        else:
            print("Continue: ",h)
            return h + self.hMB(h-1)

#Tester
t = Trace()
print("Finally ",t.hMB(5))
```

3. **[Medium]** **[7 Marks]**
Suppose, you have been given a non-negative integer which is the height of a 'house of cards'. To build such a house you at-least require 8 cards. To increase the level (or height) of that house, you would require four sides and a base for each level. Therefore, for the top level, you would require 8 cards and for each of the rest of the levels below you would require 5 extra cards. If you were asked to build level one only, you would require just 8 cards. Of course, the input can be zero; in that case, you do not build a house at all. Complete the **recursive method** below to calculate the number of cards required to build a 'house of cards' of specific height given by the parameter.



```
public int hocBuilder (int height){
      // TO DO
}
```

OR

```
def hocBuilder(height):
      #TO DO
```

4. **Draw** the **recursive flow diagram/memory stack** (draw each methods and their behavior in the way they are being called and executed) of the code given below: **[6 Marks]**

```
public class Surprise{
   public static int mystery(int n) {
     System.out.println("h(" + n + ")");
     if (n == 0) {
         System.out.println("value: 0"); return 0;
     } else {
         System.out.println("going down");
         int temp = mystery (n - 1) + 1;
         System.out.println("h(" + n + ") --> " + temp);
         return temp;
     }
   }
   public static void main(String [] args){
      mystery(4);
   }
}
```

OR

```
class Surprise:
     def mystery(self,n):
          print("h(" ,n,")")
          if(n==0):
               print("value: 0")
               return 0
          else:
               print("going down")
               temp = self.mystery(n-1)+1
               print("h(",n,") --> ",temp)
               return temp

#Tester
s = Surprise()
s.mystery(4)
```

5. **[Hard]** **[8 + 8 = 16 Marks]**
   a. Print the following pattern for the given input (**you must use recursion**):
   Sample Input: 5
   Sample Output:

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

b. Print the following pattern for the given input (**you must use recursion**):
Sample Input: 5
Sample Output:

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

6. **[Very Hard]** **[9 Marks]**
Suppose, you are working in a company 'X' where your job is to calculate the profit based on their investment.
If the company invests 100,000 USD or less, their profit will be based on 75,000 USD as first 25,000 USD goes to set up the business in the first place. For the first 100,000 USD, the profit margin is low: 4.5%. Therefore, for every 100 dollar they spend, they get a profit of 4.5 dollar.
For an investment greater than 100,000 USD, for the first 100,000 USD (actually on 75,000 USD as 25,000 is the setup cost), the profit margin is 4.5% where for the rest, it goes up to 8%. For example, if they invest 250,000 USD, they will get an 8% profit for the 150,000 USD. In addition, from the rest 100,000 USD, 25,000 is the setup cost and there will be a 4.5% profit on the rest 75,000. Investment will always be greater or equal to 25,000 and multiple of 100.
Complete the **RECURSIVE methods** below that take an array of integers (investments) and an iterator (always sets to ZERO('0') when the method is initially called) and prints the profit for the corresponding investment. You must avoid loop and multiplication ('*') operator.

```
public class FinalQ{
  public static void print(int[]array,int idx){
      if(idx<array.length){
            double profit=calcProfit(array[idx]);
      //TODO
      }
  }
  public static double calcProfit(int investment){
      //TODO
  }
  public static void main(String[]args){
     int[]array={25000,100000,250000,350000};
     print(array,0);
  }
}
```

OR

```
class FinalQ:
    def print(self,array,idx):
        if(idx<len(array)):
                profit = self.calcProfit(array[idx])
        #TO DO

    def calcProfit(self,investment):
        #TO DO

#Tester
array=[25000,100000,250000,350000]
f = FinalQ()
f.print(array,0)
```

```
-------------------------------------------------
Output:
1. Investment: 25000; Profit: 0.0
2. Investment: 100000; Profit: 3375.0
3. Investment: 250000; Profit: 15375.0
4. Investment: 350000; Profit: 23375.0
```

7. Suppose you have two arrays: Arr1 and Arr2. **[7 Marks]**
Arr1 will be sorted values. For each element v in Arr2, you need to write a pseudo code that will print the number of elements in Arr1 that is less than or equal to v.
For example: suppose you are given two arrays of size 5 and 4 respectively.
5 4 [size of the arrays]
Arr1 = 1 3 5 7 9
Arr2 = 6 4 8
The output should be 3 2 4

Explanation: Firstly, you should search how many numbers are there in Arr1 which are less than 6. There are 1, 3, 5 which are less than 6 (total 3 numbers). Therefore, the answer for 6 will be 3.
After that, you will do the same thing for 4 and 8 and output the corresponding answers which are 2 and 4. Your searching method should not take more than O (log n) time.

Sample Input	Sample Output

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

<u>BONUS TASKS: (5 MARKS)</u>

8. On a boring Friday, you are sitting alone in your room. Due to the lockdown, you cannot go out to play with your friends, so you had nothing to do at home. Suddenly, your little sister comes and tells you that some of your family members want to play an interesting game called "Friday Fun". Therefore, you decided to join in the game. There are few rules in the game:
    a. All the members that will play the game must sit in a circular manner.
    b. There will be a dice, which will determine the faith of the player.
    c. If dice after flipping get '1 or 6', the player has to pass the coin to the next player (sitting to his/her right).
    d. If dice after flipping get '3 or 5, then the player is in danger but does not get eliminated, so the player has to pass it to the next member (sitting to his/her right).
    e. If dice after flipping get '2 or 4, the player gets eliminated and passes the coin to the next player (sitting to his/her right).
    f. The game continues until there is only one member left and he/she will get a surprise gift.

   Write a method that finds out the winner of the 'Friday Fun' game, you will be given as input the number of players playing and the dice sequence. **[You must use recursion for solving this problem.] [3 Marks]**

Sample Input	Sample Output	Explanation
3, "152"	1	1st turn: Starting from 1, the dice is '1', so he passes it to the second player. 2nd turn: dice is '5', so the second player passes it to the third member. 3rd turn: dice is '2', so the third player gets eliminated and passes it to the first member. 4rd turn: dice is '1' so the first player passes it to the second player. 5th turn: dice is '5' so the second player passes it to the first player. 6th turn: dice is '2', so the second player gets eliminated and passes it to the first player. No further round is required as there is only one member so the winner is the first player.

9. Suppose an array is given **[2 Marks]**
A = [A, C, E, F, K, L, M, N, Y, Z]

a. Draw a complete **BINARY** tree from array A.
b. Write preorder, inorder, and postorder traversal for the created complete binary tree.
c. Draw a complete **TERNARY** tree from array A.
d. Write preorder and postorder traversal for the created complete ternary.
e. Draw the adjacency matrix and adjacency list for the complete ternary tree/graph. [You must consider the **tree direction from top to bottom** when drawing adjacency matrix and list]