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

a) Implement a recursive algorithm to find factorial of *n*.
b) Implement a recursive algorithm to find the *n*-th Fibonacci number.
c) Print all the elements of a given array recursively.
d) 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 each]**

a)Implement a recursive algorithm that takes a decimal number *n* and converts *n* to its corresponding (you may return as a string) binary number.

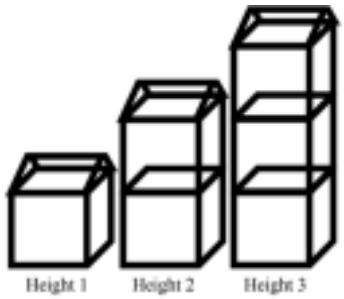
b) Implement a recursive algorithm to add all the elements of a non-dummy headed singly linked linear list. Only head of the list will be given as parameter where you may assume every node can contain only integer as its element.
Note: you'll need a Singly Node class for this code.

c) Implement a recursive algorithm which will print all the elements of a non-dummy headed singly linked linear list in reversed order.
Example: if the linked list contains 10, 20, 30 and 40, the method will print
40
30
20
10
Note: you'll need a Singly Node class for this code.

3. **[Medium] [8 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. **[Hard] [10 + 10 = 20 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

5. **[Very Hard] [12 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