Problem 1

PART A

1. For the given number of rows, write a program to print the pattern mentioned in the example without using arrays.

Example:

Input: 5

Output

```
1
2  6
3  7  10
4  8  11  13
5  9  12  14  15
```

2. Write a program to print all the LEADERS in the array. An element is a leader if it is greater than all the elements to its right side. And the rightmost element is always a leader.

Input : {16, 17, 4, 3, 5, 2}

Output : 17, 5, 2

Solution:

```
import java.util.*;

public class Prob1{
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        for(int i=0;i<n;i++){
            int v = i+1;
            int k = n-1;
```

```
            for(int j=0;j<=i;j++){
                System.out.print(v+" ");
                v += k;
                k--;
            }
            System.out.println();
        }
        sc.close();
    }
}
```

Time: O(N^2)
Space: O(1)

Problem 2:

2. Given an unsorted array of integers, find the largest contiguous pair sum in it.

Sample input 1:
Enter the array size : 8
Input : 0 5 6 0 9 0 0 1
Output : 11
Explanation : 5 and 6 form the largest sum.

Sample input 2:
Enter the array size : 5
Input : 12 31 10 6 40
Output : 46
Explanation 6 and 40 form the largest sum.

Sample input 3:
Enter the array size : 4
Input :3 5 4 4
Output : 9

Solution:

```
import java.util.*;

public class Prob5 {
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int arr[] = new int[n];
        for(int i=0;i<n;i++) arr[i] = sc.nextInt();
        int ans = 0;
        for(int i=0;i<n-1;i++){
            int curr = arr[i]+arr[i+1];
```

```
            ans = Math.max(ans,curr);
        }
        System.out.println(ans);
        sc.close();
    }
}
```

TIme: O(n)
Space: O(1)

Problem 3:

3. Find the number x in a matrix. x is defined as the minimum number in a row and maximum number in the same column. Print -1 if there is no such element.

Test case 1

Enter the array size : 3 3
input array : 3 11 7
5 9 14
2 4 5
output : 5
Explanation : 5 is the minimum number in the row [5 9 14] and maximum number in the column [3 5 2]

Test case 2

Enter the array size : 4 2
input array : 4 11
9 8
3 27
7 7
output: -1
Explanation : There is no matching number for the given input.

Test case 3

Enter the array size : 3 4
nput array : 4 7 0 13
8 5 3 1
42 23 4 9
utput : 4
xplanation : 4 is the minimum number in the row [42 23 4 9] and maximum number
the column [0 3 4]

Solution:

```
import java.util.*;
```

```java
public class Prob3 {
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int m = sc.nextInt();
        int arr[][] = new int[n][m];
        for(int i=0;i<n;i++){
            for(int j=0;j<m;j++) arr[i][j] = sc.nextInt();
        }
        for(int i=0;i<n;i++){
            for(int j=0;j<m;j++){
                if(isValid(i,j,arr,n,m)){
                    System.out.println(arr[i][j]);
                    sc.close();
                    return;
                }
            }
        }
        System.out.println(-1);
        sc.close();
    }
    public static boolean isValid(int i,int j,int arr[][],int n,int m){
        for(int c=0;c<m;c++){
            if(c==j) continue;
            if(arr[i][c]<arr[i][j]) return false;
        }
        for(int r=0;r<n;r++){
            if(r==i) continue;
            if(arr[r][j]>arr[i][j]) return false;
        }
        return true;
    }
}
```

Time: O(n*m)
Space O(1)

Problem 4:

1. Given positive values for A and an array of size N , print the sum of the indexes that contain the given digit A

Note: Digit A is a single digit.

Test cases :

Case 1:

A=6

N=4

Array : 106,12,13,16

Output : 3

Explanation : Indexes of values that contain 6 are 0 & 3, their sum is 3.

Case 2:

A=2

N=6

Array: 1,3,124,5,6,12

Output : 7

Case 3:

A=1

N=5

Array: 1,2,3,51,6

Output : 3

Case 3:

A=2

N=5

Array: 111,222,203,5201,6

Output : 6

Solution:

```
import java.util.*;

public class Prob4 {
```

```
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
        int n = sc.nextInt();
        int sum = 0;
        for(int i=0;i<n;i++){
            int v = sc.nextInt();
            while(v>0){
                int d = v%10;
                if(d==a){
                    sum += i;
                    break;
                }
                v = v/10;
            }
        }
        System.out.println(sum);
        sc.close();
    }
}
```

Time: O(n)
Space: O(1)

Problem 5:

PART A

1. For the given number of rows, write a program to print the pattern mentioned in the example without using arrays.

Example:

Input: 5

Output

```
1
2  6
3  7  10
4  8  11  13
5  9  12  14  15
```

2. Write a program to print all the LEADERS in the array. An element is a leader if it is greater than all the elements to its right side. And the rightmost element is always a leader.

Input : {16, 17, 4, 3, 5, 2}

Output : 17, 5, 2

Solution:

```java
import java.util.*;

public class Prob2 {
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int arr[] = new int[n];
        for(int i=0;i<n;i++) arr[i] = sc.nextInt();
        int max = -1;
        List<Integer> list = new ArrayList<>();
        for(int i=n-1;i>=0;i--){
            if(arr[i]>max) list.add(arr[i]);
```

```
            max = Math.max(max,arr[i]);
        }
        Collections.reverse(list);
        for(int x:list){
            System.out.print(x+" ");
        }
        System.out.println();
        sc.close();
    }
}
```

Time: O(n)
Space: O(n)

Problem 6:

4.Find maximum Sum of Two Non-Overlapping Subarrays

Given an unsorted array A[] of non-negative integers, return the maximum sum of elements in two non-overlapping (contiguous) subarrays, which have lengths L and M.

The L-length subarray could occur before or after the M-length subarray

Test case 1:
Enter the array size: 9
Input array: [0,6,5,2,2,5,1,9,4] , L = 1, M = 2
Output: 20
Explanation: One choice of subarrays is [9] with length 1, and [6,5] with length 2.

Test case 2:
Enter the array size: 9
Input array: [3,8,2,3,2,1,8,9,0], L = 3, M = 2
Output: 30
Explanation: One choice of subarrays is [3,8,2] with length 3, and [8,9] with length 2.

Test case 3:
Input array: [2,1,2,3,5,6,1,0,8,9], L = 4, M = 2
Output: 33
Explanation: One choice of subarrays is [2,3,5,6] with length 4, and [8,9] with length

Solution:

```
import java.util.*;

public class Main {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int l = sc.nextInt();
        int m = sc.nextInt();
        int[] arr = new int[n];

        for(int i=0;i<n;i++){
```

```
            arr[i] = sc.nextInt();
        }
        System.out.println(Math.max(f(arr,l,m),f(arr,m,l)));

    }

    public static int f(int[] arr, int l, int m) {
        int ans = 0;
        int n = arr.length;
        int[] pref = new int[n+1];
        for(int i=0;i<n;i++){
            pref[i+1] = pref[i] + arr[i];
        }
        int maxl = 0;
        for(int i=l+m;i<=n;i++){
            maxl = Math.max(maxl,pref[i-m]-pref[i-m-l]);
            int msum = pref[i] - pref[i-m];
            ans = Math.max(ans,maxl+msum);
        }
        return ans;
    }
}
```

Time: O(n)
Space: O(n)