EXPERIMENT - 1.2

Name: Thirumurugan V G | UID: 21BCS4984

Branch: CSE | Section: NTPP_IOT_602-B

Semester: 5th | Date: 17-08-2023

Subject: AP Lab | Subject Code: 21-CSP-314

1) Aim: To implement the concept of Stacks and Queues.

- **Problem1:** https://www.hackerrank.com/challenges/equal-stacks/problem
- **Problem2:** https://www.hackerrank.com/challenges/game-of-two-stacks/problem

2) Program Code:

Problem1: You have three stacks of cylinders where each cylinder has the same diameter, but they may vary in height. You can change the height of a stack by removing and discarding its topmost cylinder any number of times. Find the maximum possible height of the stacks such that all of the stacks are exactly the same height. This means you must remove zero or more cylinders from the top of zero or more of the three stacks until they are all the same height, then return the height.

Code:

```
import java.io.*;
import java.util.*;
public class Solution {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        HashMap<Integer, ArrayList<Integer>> map = new HashMap<Integer, Ar
rayList<Integer>>();
```

```
        int[] sum = new int[3];
        int[] sn = new int[3];
        for (int i =0; i< 3 ; i++)
        {
           sn[i] = in.nextInt();
        }
        for(int i =0; i < 3; i++)
        {
           map.put(i, new ArrayList<Integer>());
           for(int j=0 ; j < sn[i]; j++)
           {
              int k = in.nextInt();
              sum[i] += k;
              map.get(i).add(0,k);
           }
        }
        while (!((sum[0] == sum[1]) && (sum[1] == sum[2])))
        {
           int minSum = Math.max(Math.max(sum[0], sum[1]), sum[2]);
           int j = 0;
           if (minSum == sum[0])
              j =0;
           else if (minSum == sum[1])
              j = 1;
           else
              j = 2;
              sum[j] -=  map.get(j).get(map.get(j).size() - 1);
              map.get(j).remove(map.get(j).size() - 1);
        }
        System.out.println(sum[0]);
    }
}
```

Problem2: Alexa has two stacks of non-negative integers, stack a[n] and stack b[m] where index 0 denotes the top of the stack. Alexa challenges Nick to play the following game:

- In each move, Nick can remove one integer from the top of either stack a or stack b.
- Nick keeps a running sum of the integers he removes from the two stacks.

- Nick is disqualified from the game if, at any point, his running sum becomes greater than some integer maxSum given at the beginning of the game.
- Nick's *final score* is the total number of integers he has removed from the two stacks.

Given a, b, and maxSum for g games, find the maximum possible score Nick can achieve.

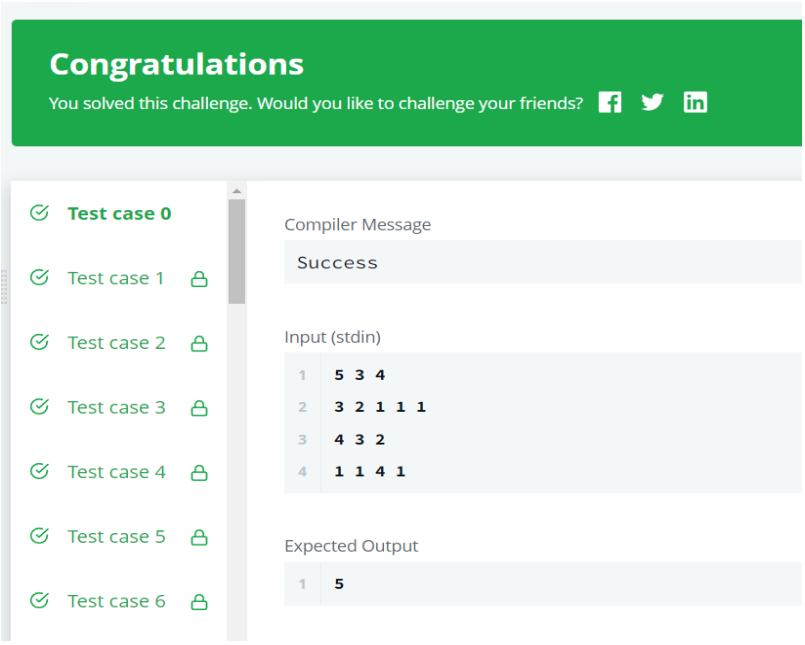
Code:

```
import java.util.*;
public class GameOfTwoStacks {
   public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      int g = in.nextInt();
      for(int a0 = 0; a0 < g; a0++){
         int n = in.nextInt();
         int m = in.nextInt();
         int x = in.nextInt();
         long[] a_sum = new long[n];
         a_sum[0] = in.nextLong();
         for(int a_i=1; a_i < n; a_i++){

            a_sum[a_i] = in.nextLong()+a_sum[a_i-1];
         }
         long[] b_sum = new long[m];
         b_sum[0] = in.nextLong();

         for(int b_i=1; b_i < m; b_i++){
            b_sum[b_i] = in.nextLong()+b_sum[b_i-1];
         }
         int ai = a_sum.length-1;
         int bi = 0;
         int max_score = 0;
         int score = 0;
         while(ai>=0&&a_sum[ai] > x)
            ai--;
         if(ai>=0)
            while(bi<b_sum.length&&a_sum[ai]+b_sum[bi]<=x)
               bi++;
         else
            while(bi<b_sum.length&&b_sum[bi]<=x)
```

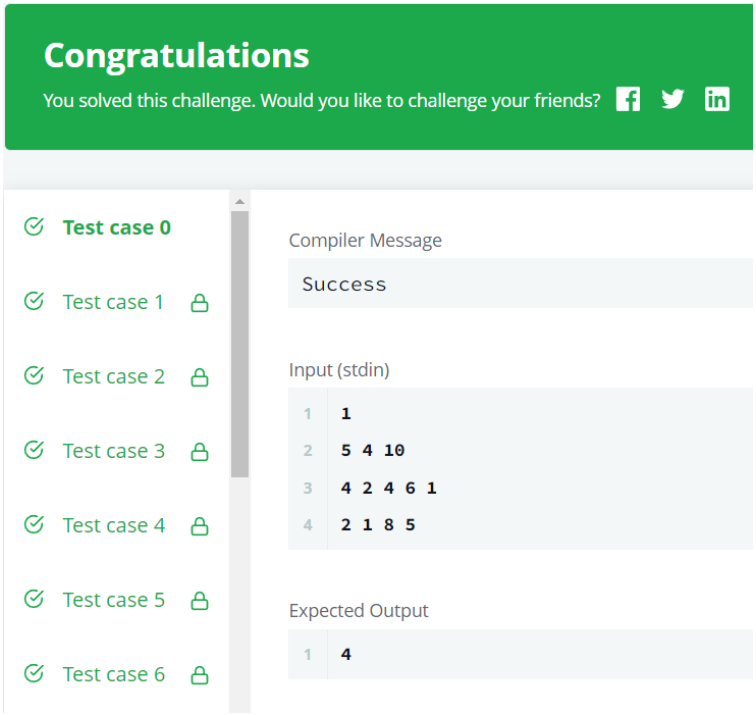
```
            bi++;
        for(ai = ai; ai >= -1; ai--)
        {
            if(ai>=0)
            {
                while(bi<b_sum.length&&a_sum[ai]+b_sum[bi]<=x)
                    bi++;
                score = ai+bi+1;
            }
            else
            {
                while(bi<b_sum.length&&b_sum[bi]<=x)
                    bi++;
                score = bi;
            }
            if(score>max_score)
                max_score = score;
        }
        System.out.println(max_score);
    }
  }
}
```

3) Output:

1.



2.



4) Learning Outcomes:

- The logic to find out the solution and achieve all test cases.
- Working of the concepts like HashMap and Arraylist.
- Understand the implementation of Stacks.