Experiment1.2

Student Name: Nishant Kumar mehta **UID: 21BCS3402**
Branch: CSE **Section/Group: IOT - 602**
Semester: 5th **Dateof Performance: 18/08/23**
Subject Name: Advance Programming Lab **Subject Code: 21CSP - 314**

1. Aim: To implement the concept of Stack and Queues.

2. Objective:

1. 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.
2. 2. Alexa has two stacks of non-negative integers, stack and stack where index 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 or stack .
   • 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 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 , , and for games, find the maximum possible score Nick can achieve.

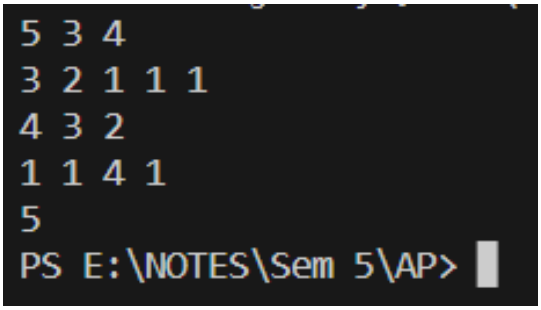
3. Program and output:

1.

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;public class Solution {public static void main(String[] args) {
 Scanner in = new Scanner(System.in);
```

```
int n1 = in.nextInt();
int n2 = in.nextInt();
int n3 = in.nextInt();
int h1[] = new int[n1];
int sum1 = 0;
int sum2 = 0;
int sum3 = 0;
for(int h1_i=0; h1_i < n1; h1_i++){
h1[h1_i] = in.nextInt();
sum1 += h1[h1_i];
}
int h2[] = new int[n2];
for(int h2_i=0; h2_i < n2; h2_i++){
h2[h2_i] = in.nextInt();
sum2 += h2[h2_i];
}
int h3[] = new int[n3];
for(int h3_i=0; h3_i < n3; h3_i++){
h3[h3_i] = in.nextInt();
sum3 += h3[h3_i];
}
boolean equal = false;
int index1 = 0;
int index2 = 0;
int index3 = 0;
while(!equal){
if((sum1 == sum2) && ( sum1 == sum3))
equal = true;
else{
if(sum1 > sum2){
if(sum1 > sum3){
sum1 -= h1[index1];
index1++;
}else{
sum3 -=h3[index3];
index3++;
}
}else{
if(sum2 > sum3){
sum2 -= h2[index2];
index2++;
```

```
}else{
sum3 -= h3[index3];
index3++;
}
}
}
}
System.out.println(sum1);
}
}
```

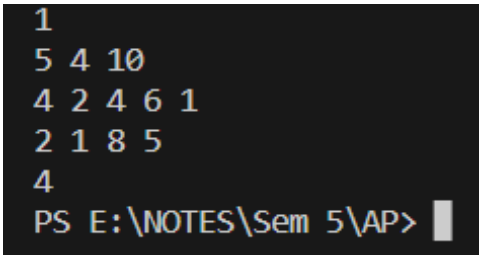


2.

```
import java.util.Scanner;

public class GameOfTwoStacks {

    static int twoStacks(int x, int[] a, int[] b) {

        int a_index = 0;
        int b_index = 0;
        int count = 0;
        int sum = 0;
        // move b_index to the position where if only take elements from B, last element it can take
        while (b_index < b.length && sum + b[b_index] <= x) {
            sum += b[b_index];
            b_index++;
        }
        // loop exits only when b_index reaches end or sum > x; in both case b_index should decrease
        b_index--;
        count = b_index + 1;
```

```
    while (a_index < a.length && b_index < b.length) {
        sum += a[a_index];
        if (sum > x) {
            while (b_index >= 0) {
                sum -= b[b_index];
                b_index--;
                if (sum <= x) break;
            }
            // if even no elements taken from B, but still sum greater than x, then a[a_index] should not be chosen
            // and loop terminates
            if (sum > x && b_index < 0) {
                a_index--;
                break;
            }
        }
        count = Math.max(a_index + b_index + 2, count);
        a_index++;
    }

    return count;
}

private static final Scanner scanner = new Scanner(System.in);

public static void main(String[] args) {

    int g = Integer.parseInt(scanner.nextLine().trim());

    for (int gItr = 0; gItr < g; gItr++) {

        String[] nmx = scanner.nextLine().split(" ");

        int n = Integer.parseInt(nmx[0].trim());
        int m = Integer.parseInt(nmx[1].trim());
        int x = Integer.parseInt(nmx[2].trim());

        int[] a = new int[n];
        String[] aItems = scanner.nextLine().split(" ");
        for (int aItr = 0; aItr < n; aItr++) {
            int aItem = Integer.parseInt(aItems[aItr].trim());
            a[aItr] = aItem;
        }

        int[] b = new int[m];
        String[] bItems = scanner.nextLine().split(" ");
        for (int bItr = 0; bItr < m; bItr++) {
            int bItem = Integer.parseInt(bItems[bItr].trim());
            b[bItr] = bItem;
        }

        int result = twoStacks(x, a, b);
        System.out.println(result);
```

```
        }
    }
}
```


1
5 4 10
4 2 4 6 1
2 1 8 5
4
PS E:\NOTES\Sem 5\AP>