S.Ramyabharathi_AIDS_practiseset1

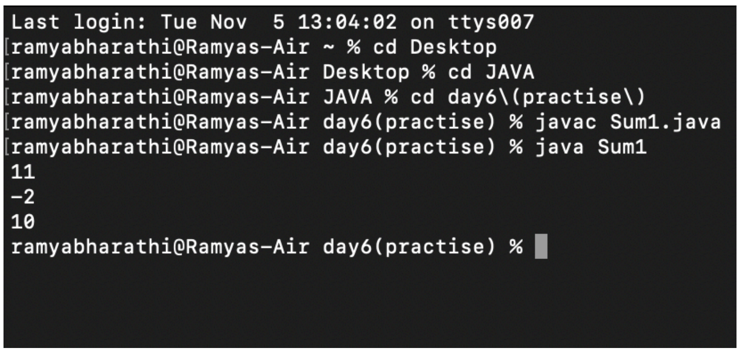
1.

Maximum Subarray Sum – Kadane‟s Algorithm:
Given an array arr[], the task is to find the subarray that has the maximum sum and return its sum.
Input: arr[] = {2, 3, -8, 7, -1, 2, 3}
Output: 11
Explanation: The subarray {7, -1, 2, 3} has the largest sum 11.
Input: arr[] = {-2, -4}
Output: –2
Explanation: The subarray {-2} has the largest sum -2.
Input: arr[] = {5, 4, 1, 7, 8}
Output: 25
Explanation: The subarray {5, 4, 1, 7, 8} has the largest sum 25.

Solution:

```java
import java.util.*;
class Sum1{
   int Maxsubarr(int[] nums){
       int r=nums[0];
       for(int i=0;i<nums.length;i++){
           int s=0;
           for(int j=i;j<nums.length;j++){
               s=s+nums[j];
               r=Math.max(r,s);
           }
       }
       return r;
   }
   public static void main(String[] args){
       int[] nums={2, 3, -8, 7, -1, 2, 3};
       int[] nums1={-2, -4};
       int[] nums2={4,2,-1,5};
       Sum1 p=new Sum1();
       System.out.println(p.Maxsubarr(nums));
       System.out.println(p.Maxsubarr(nums1));
       System.out.println(p.Maxsubarr(nums2));
   }
}
```

Output:



```
Last login: Tue Nov  5 13:04:02 on ttys007
[ramyabharathi@Ramyas-Air ~ % cd Desktop
[ramyabharathi@Ramyas-Air Desktop % cd JAVA
[ramyabharathi@Ramyas-Air JAVA % cd day6\(practise\)
[ramyabharathi@Ramyas-Air day6(practise) % javac Sum1.java
[ramyabharathi@Ramyas-Air day6(practise) % java Sum1
11
-2
10
ramyabharathi@Ramyas-Air day6(practise) % 
```

Time Complexity:O(n^2)

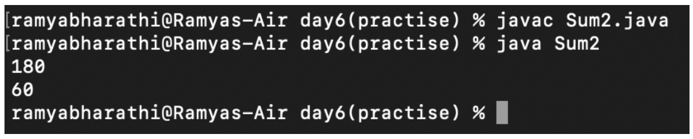
2.

Maximum Product Subarray
Given an integer array, the task is to find the maximum product of any subarray.
Input: arr[] = {-2, 6, -3, -10, 0, 2}
Output: 180
Explanation: The subarray with maximum product is {6, -3, -10} with product = 6 * (-3) * (-10) = 180
Input: arr[] = {-1, -3, -10, 0, 60}
Output: 60
Explanation: The subarray with maximum product is {60}.

Solution:

```java
import java.util.*;

class Sum2 {
   int maxp(int[] nums) {
       if (nums.length == 0) return 0;
       int maxp = nums[0];
       int minP = nums[0];
       int r = nums[0];
       for (int i = 1; i < nums.length; i++) {
           int c = nums[i];
           if (c < 0) {
               int temp = maxp;
               maxp = minP;
               minP = temp;
           }
           maxp = Math.max(c,maxp*c);
           minP = Math.min(c,minP*c);
           r = Math.max(r, maxp);
       }

       return r;
   }

   public static void main(String[] args) {
       int[] nums = {-2, 6, -3, -10, 0, 2};
       int[] nums1 =  {-1, -3, -10, 0, 60};
       Sum2 p = new Sum2();
       System.out.println(p.maxp(nums));
       System.out.println(p.maxp(nums1));
   }
}
```

Output:



```
[ramyabharathi@Ramyas-Air day6(practise) % javac Sum2.java
[ramyabharathi@Ramyas-Air day6(practise) % java Sum2
180
60
ramyabharathi@Ramyas-Air day6(practise) %
```

Time Complexity:O(n)

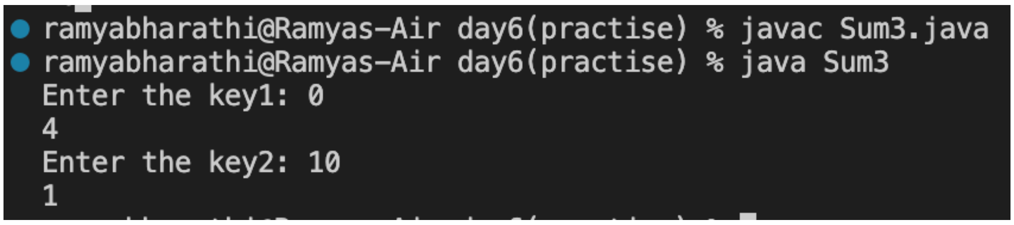
3.

Search in a sorted and rotated Array
Given a sorted and rotated array arr[] of n distinct elements, the task is to find the index of given key in the array. If the key is not present in the array, return -1.
Input : arr[] = {4, 5, 6, 7, 0, 1, 2}, key = 0
Output : 4
Input : arr[] = { 4, 5, 6, 7, 0, 1, 2 }, key = 3
Output : -1
Input : arr[] = {50, 10, 20, 30, 40}, key = 10
Output : 1

Solution:

```java
import java.util.*;
class Sum3{
   int Arrinx(int[] nums, int key){

       for(int i=0;i<nums.length;i++){
           if (nums[i]==key){
               return i;
           }

       }
       return -1;
   }
   public static void main(String[] args){
       int[] nums={4, 5, 6, 7, 0, 1, 2};
       int[] nums1={50, 10, 20, 30, 40};
       Sum3 p=new Sum3();
       Scanner sc= new Scanner(System.in);
       System.out.print("Enter the key1: ");
       int key1=sc.nextInt();
       System.out.println(p.Arrinx(nums,key1));
       System.out.print("Enter the key2: ");
       int key2=sc.nextInt();
       System.out.println(p.Arrinx(nums1,key2));


   }
}
```

Output:

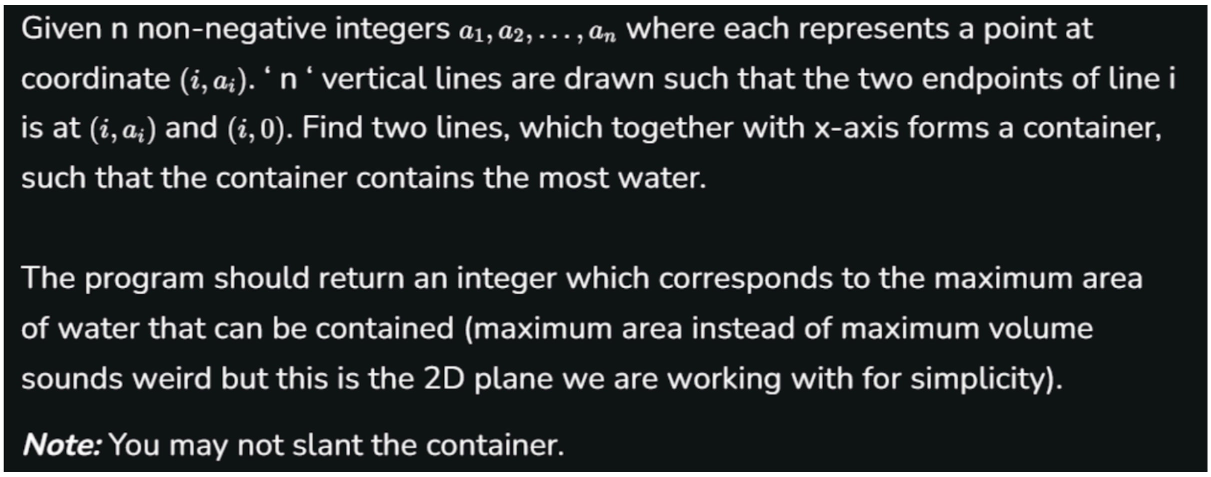


```
ramyabharathi@Ramyas-Air day6(practise) % javac Sum3.java
ramyabharathi@Ramyas-Air day6(practise) % java Sum3
Enter the key1: 0
4
Enter the key2: 10
1
```

Time Complexity:O(n)

4.

Container with Most Water



Given n non-negative integers $a_1, a_2, \ldots, a_n$ where each represents a point at coordinate (i, a_i). ‘ n ‘ vertical lines are drawn such that the two endpoints of line i is at (i, a_i) and $(i, 0)$. Find two lines, which together with x-axis forms a container, such that the container contains the most water.

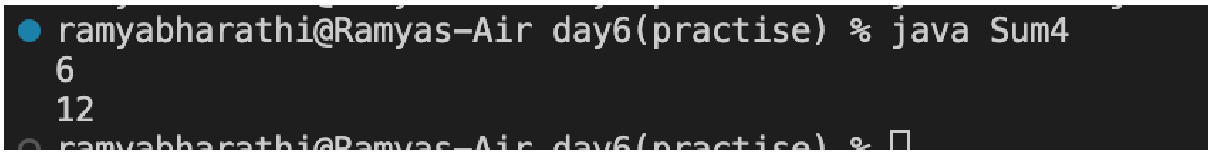
The program should return an integer which corresponds to the maximum area of water that can be contained (maximum area instead of maximum volume sounds weird but this is the 2D plane we are working with for simplicity).

Note: You may not slant the container.

Solution:

```java
import java.util.*;
class Sum4{
   int Arrinx(int[] nums){
       int fa=0;
       int h=0;
       int b=0;
       for (int i=0;i<nums.length;i++){
           for (int j=i;j<nums.length;j++){
               b=j-i;
               h=Math.min(nums[i],nums[j]);
               fa=Math.max(fa,b*h);
           }
       }
       return fa;
   }
   public static void main(String[] args){
       int[] nums={1, 5, 4, 3};
       int[] nums1={3, 1, 2, 4, 5};
       Sum4 p=new Sum4();
       System.out.println(p.Arrinx(nums));
       System.out.println(p.Arrinx(nums1));

}
```

Output:



```
ramyabharathi@Ramyas-Air day6(practise) % java Sum4
6
12
ramyabharathi@Ramyas-Air day6(practise) %
```

Time Complexity:O(n^2)

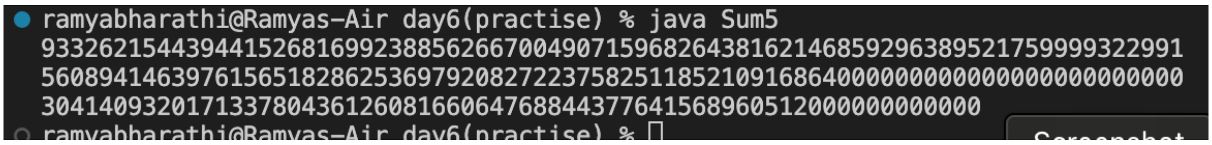
5.

Find the Factorial of a large number
Input: 100
Output:
9332621544394415268169923885626670049071596826438162146859296389521759999322991
5608941463976156518286253697920827223758251185210916864000000000000000000000000
00
Input: 50
Output: 30414093201713378043612608166064768844377641568960512000000000000

Solution:

```java
import java.math.BigInteger;
import java.util.Scanner;

public class Sum5 {
   static BigInteger factorial(int n)
   {
       BigInteger fact= new BigInteger("1");
       for (int i = 2; i <= n; i++)
           fact = fact.multiply(BigInteger.valueOf(i));

       return fact;
   }
   public static void main(String args[]) throws Exception
   {
       int n = 100;
       int n1=50;
       System.out.println(factorial(n));
       System.out.println(factorial(n1));
   }
}
```

Output:



```
ramyabharathi@Ramyas-Air day6(practise) % java Sum5
9332621544394415268169923885626670049071596826438162146859296389521759999322991
5608941463976156518286253697920827223758251185210916864000000000000000000000000
30414093201713378043612608166064768844377641568960512000000000000
ramyabharathi@Ramyas-Air day6(practise) %
```

Time Complexity:O(n)

6.

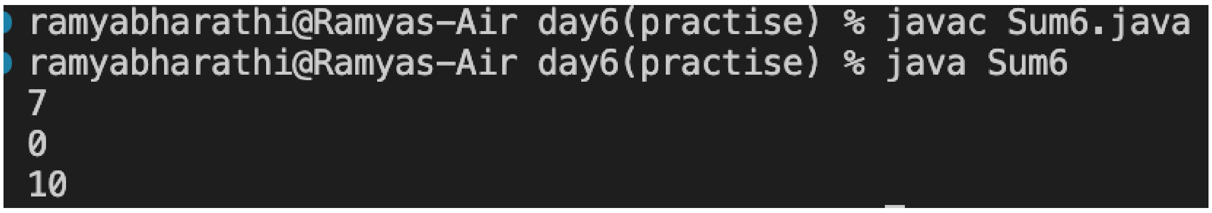
Trapping Rainwater Problem states that given an array of n non-negative integers arr[] representing an elevation map where the width of each bar is 1, compute how much water it can trap after rain.
Input: arr[] = {3, 0, 1, 0, 4, 0, 2}
Output: 10
Explanation: The expected rainwater to be trapped is shown in the above image.
Input: arr[] = {3, 0, 2, 0, 4}
Output: 7
Explanation: We trap 0 + 3 + 1 + 3 + 0 = 7 units.
Input: arr[] = {1, 2, 3, 4}
Output: 0
Explanation : We cannot trap water as there is no height bound on both sides
Input: arr[] = {10, 9, 0, 5}
Output: 5
Explanation : We trap 0 + 0 + 5 + 0 = 5

Solution:

```
import java.util.*;
class Sum6{
   int Waterfilled(int[] nums){
       int l=0;
       int r=nums.length-1;
       int lmax=0;
       int rmax=0;
       int water=0;
       while(l<r){
           if (nums[l]<nums[r]){
               if (nums[l]>=lmax){
                   lmax+=nums[l];
               }
               else{
                   water+=lmax-nums[l];
               }
               l++;
           }
           else{
               if(nums[r]>=rmax){
                   rmax+=nums[r];
               }
               else{
                   water+=rmax-nums[r];
               }
               r--;
           }
       }
       return water;
   }

   public static void main(String[] args){
       int[] nums={3, 0, 2, 0, 4};
       int[] nums1={1, 2, 3, 4};
       int[] nums2={3, 0, 1, 0, 4, 0, 2};
```

```
        Sum6 p=new Sum6();
        System.out.println(p.Waterfilled(nums));
        System.out.println(p.Waterfilled(nums1));
        System.out.println(p.Waterfilled(nums2));
    }
}
```

Output:



```
ramyabharathi@Ramyas-Air day6(practise) % javac Sum6.java
ramyabharathi@Ramyas-Air day6(practise) % java Sum6
7
0
10
```

Time Complexity:O(1)

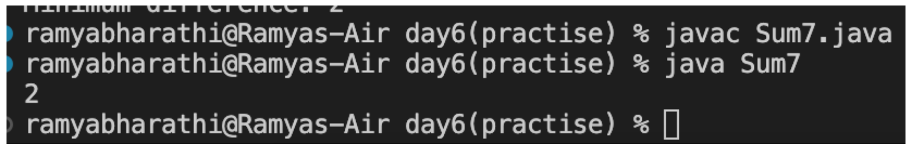
7.

Chocolate Distribution Problem
Given an array arr[] of n integers where arr[i] represents the number of chocolates in ith packet.
Each packet can have a variable number of chocolates. There are m students, the task is to distribute chocolate packets such that:
Each student gets exactly one packet.
The difference between the maximum and minimum number of chocolates in the packets given to the students is minimized.
Input: arr[] = {7, 3, 2, 4, 9, 12, 56}, m = 3
Output: 2
Explanation: If we distribute chocolate packets {3, 2, 4}, we will get the minimum difference, that is 2.
Input: arr[] = {7, 3, 2, 4, 9, 12, 56}, m = 5
Output: 7
Explanation: If we distribute chocolate packets {3, 2, 4, 9, 7}, we will get the minimum difference, that is 9 – 2 = 7.

Solution:

```
import java.util.Arrays;

class Sum7 {
    int Mindiff(int[] nums, int m) {
        if (m == 0 || nums.length < m) {
            return 0;
        }
        Arrays.sort(nums);
        int minDiff = Integer.MAX_VALUE;
        for (int i = 0; i <= nums.length - m; i++) {
            int diff = nums[i + m - 1] - nums[i];
            minDiff = Math.min(minDiff, diff);
        }
        return minDiff;
    }
    public static void main(String[] args) {
        int[] nums = {7, 3, 2, 4, 9, 12, 56};
        int m = 3;
        Sum7 p = new Sum7();
        System.out.println(p.Mindiff(nums, m));
    }
}
```

Output:



```
Minimum difference: 2
ramyabharathi@Ramyas-Air day6(practise) % javac Sum7.java
ramyabharathi@Ramyas-Air day6(practise) % java Sum7
2
ramyabharathi@Ramyas-Air day6(practise) % 
```

Time Complexity:O(nlogn)

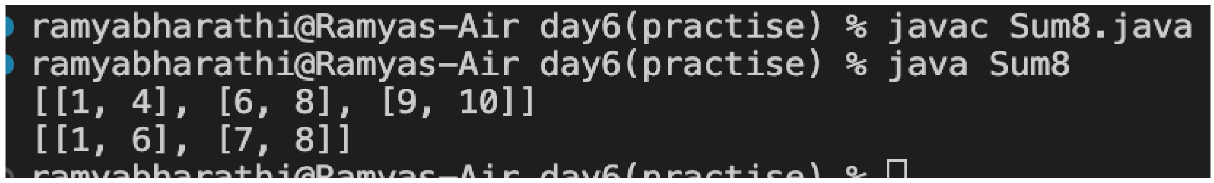
8.

Merge Overlapping Intervals
Given an array of time intervals where arr[i] = [starti, endi], the task is to merge all the overlapping intervals into one and output the result which should have only mutually exclusive intervals.
Input: arr[] = [[1, 3], [2, 4], [6, 8], [9, 10]]
Output: [[1, 4], [6, 8], [9, 10]]
Explanation: In the given intervals, we have only two overlapping intervals [1, 3] and [2, 4].
Therefore, we will merge these two and return [[1, 4}], [6, 8], [9, 10]].Input: arr[] = [[7, 8], [1, 5], [2, 4], [4, 6]]
Output: [[1, 6], [7, 8]]
Explanation: We will merge the overlapping intervals [[1, 5], [2, 4], [4, 6]] into a single interval [1, 6].

Solution:

```
import java.util.*;
class Sum8 {
   public int[][] mi(int[][] iv) {
        if (iv.length <= 1) return iv;
        Arrays.sort(iv, (a, b) -> Integer.compare(a[0], b[0]));
        List<int[]> m = new ArrayList<>();
        int[] ci = iv[0];
        m.add(ci);
        for (int[] i : iv) {
            int cs = ci[0];
            int ce = ci[1];
            int ns = i[0];
            int ne = i[1];
            if (ce >= ns) {
                ci[1] = Math.max(ce, ne);
            } else {
                ci = i;
                m.add(ci);
            }
        }
        return m.toArray(new int[m.size()][]);
   }
   public static void main(String[] args) {
        Sum8 s = new Sum8();
        int[][] arr1 = {{1, 3}, {2, 4}, {6, 8}, {9, 10}};
        int[][] arr2 = {{7, 8}, {1, 5}, {2, 4}, {4, 6}};
        System.out.println(Arrays.deepToString(s.mi(arr1)));
        System.out.println(Arrays.deepToString(s.mi(arr2)));
    }
}
```

Output:



```
ramyabharathi@Ramyas-Air day6(practise) % javac Sum8.java
ramyabharathi@Ramyas-Air day6(practise) % java Sum8
[[1, 4], [6, 8], [9, 10]]
[[1, 6], [7, 8]]
ramyabharathi@Ramyas-Air day6(practise) % 
```

Time Complexity:O(nlogn)

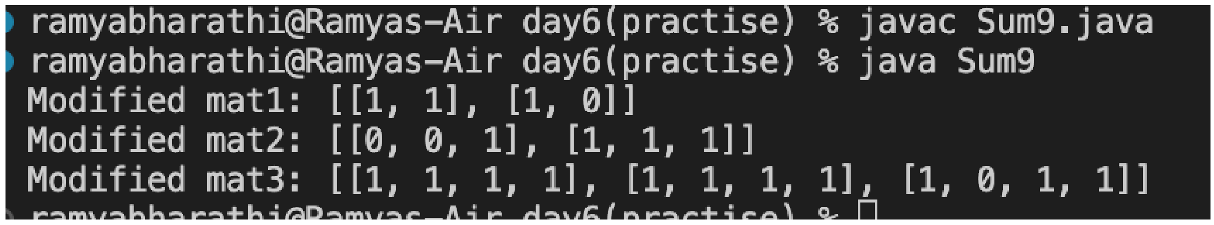
9.

A Boolean Matrix Question
Given a boolean matrix mat[M][N] of size M X N, modify it such that if a matrix cell mat[i][j] is 1 (or true) then make all the cells of ith row and jth column as 1.
Input: {{1, 0},
{0, 0}}
Output: {{1, 1}
{1, 0}}
Input: {{0, 0, 0},
{0, 0, 1}}
Output: {{0, 0, 1},
{1, 1, 1}}
Input: {{1, 0, 0, 1},
{0, 0, 1, 0},
{0, 0, 0, 0}}
Output: {{1, 1, 1, 1},
{1, 1, 1, 1},
{1, 0, 1, 1}}

Solution:

```
import java.util.*;
class Sum9 {
    public void bm(int[][] m) {
        int r = m.length, c = m[0].length;
        boolean[] rs = new boolean[r];
        boolean[] cs = new boolean[c];
        for (int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                if (m[i][j] == 1) {
                    rs[i] = true;
                    cs[j] = true;
                }
            }
        }
        for (int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                if (rs[i] || cs[j]) {
                    m[i][j] = 1;
                }
            }
        }
    }
    public static void main(String[] args) {
        Sum9 s = new Sum9();
        int[][] mat1 = {{1, 0}, {0, 0}};
        int[][] mat2 = {{0, 0, 0}, {0, 0, 1}};
        int[][] mat3 = {{1, 0, 0, 1}, {0, 0, 1, 0}, {0, 0, 0, 0}};
        s.bm(mat1);
        s.bm(mat2);
        s.bm(mat3);
        System.out.println("Modified mat1: " + Arrays.deepToString(mat1));
        System.out.println("Modified mat2: " + Arrays.deepToString(mat2));
        System.out.println("Modified mat3: " + Arrays.deepToString(mat3));
    }
}
```

Output:



```
ramyabharathi@Ramyas-Air day6(practise) % javac Sum9.java
ramyabharathi@Ramyas-Air day6(practise) % java Sum9
Modified mat1: [[1, 1], [1, 0]]
Modified mat2: [[0, 0, 1], [1, 1, 1]]
Modified mat3: [[1, 1, 1, 1], [1, 1, 1, 1], [1, 0, 1, 1]]
ramyabharathi@Ramyas-Air day6(practise) %
```

Time Complexity:O(m*n)

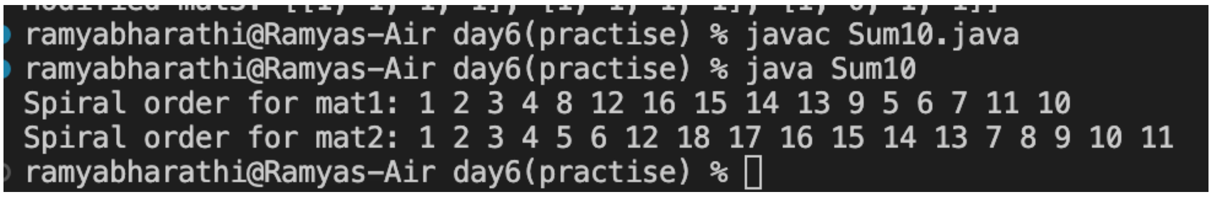
10.

Print a given matrix in spiral form
Given an m x n matrix, the task is to print all elements of the matrix in spiral form.
Input: matrix = {{1, 2, 3, 4},
{5, 6, 7, 8},
{9, 10, 11, 12},
{13, 14, 15, 16 }}
Output: 1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
Input: matrix = { {1, 2, 3, 4, 5, 6},
{7, 8, 9, 10, 11, 12},
{13, 14, 15, 16, 17, 18}}
Output: 1 2 3 4 5 6 12 18 17 16 15 14 13 7 8 9 10 11
Explanation: The output is matrix in spiral format.

Solution:

```java
import java.util.*;
class Sum10 {
   public void sp(int[][] m) {
       int r1 = 0, r2 = m.length - 1;
       int c1 = 0, c2 = m[0].length - 1;
       while (r1 <= r2 && c1 <= c2) {
           for (int i = c1; i <= c2; i++) System.out.print(m[r1][i] + " ");
           r1++;
           for (int i = r1; i <= r2; i++) System.out.print(m[i][c2] + " ");
           c2--;
           if (r1 <= r2) {
               for (int i = c2; i >= c1; i--) System.out.print(m[r2][i] + " ");
               r2--;
           }
           if (c1 <= c2) {
               for (int i = r2; i >= r1; i--) System.out.print(m[i][c1] + " ");
               c1++;
           }
       }
   }

   public static void main(String[] args) {
       Sum10 s = new Sum10();
       int[][] mat1 = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}, {13, 14, 15, 16}};
       int[][] mat2 = {{1, 2, 3, 4, 5, 6}, {7, 8, 9, 10, 11, 12}, {13, 14, 15, 16, 17, 18}};
       System.out.print("Spiral order for mat1: ");
       s.sp(mat1);
       System.out.println();
       System.out.print("Spiral order for mat2: ");
       s.sp(mat2);
       System.out.println();
   }
}
```

Output:



```
ramyabharathi@Ramyas-Air day6(practise) % javac Sum10.java
ramyabharathi@Ramyas-Air day6(practise) % java Sum10
Spiral order for mat1: 1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
Spiral order for mat2: 1 2 3 4 5 6 12 18 17 16 15 14 13 7 8 9 10 11
ramyabharathi@Ramyas-Air day6(practise) %
```

Time Complexity:O(m*n)

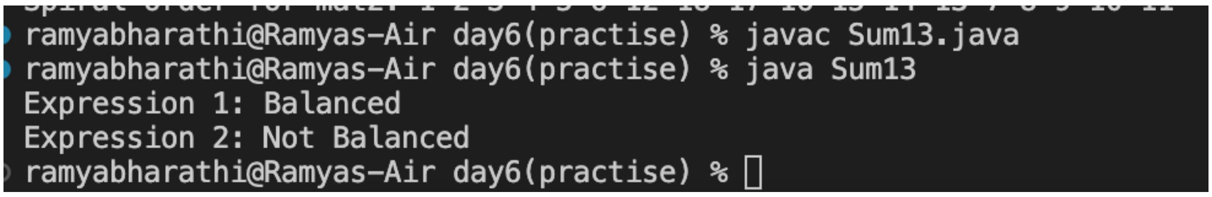
13.

Check if given Parentheses expression is balanced or not
Given a string str of length N, consisting of „(„ and „)„ only, the task is to check whether it is balanced or not.Input: str = “((()))()()”
Output: Balanced
Input: str = “())((())”
Output: Not Balanced

Solution:

```java
class Sum13 {
   public String checkBalance(String str) {
       int count = 0;
       for (char ch : str.toCharArray()) {
           if (ch == '(') {
               count++;
           } else if (ch == ')') {
               count--;
           }
           if (count < 0) {
               return "Not Balanced";
           }
       }
       return (count == 0) ? "Balanced" : "Not Balanced";
   }

   public static void main(String[] args) {
       Sum13 s = new Sum13();
       String str1 = "((()))()()";
       String str2 = "())((())";
       System.out.println("Expression 1: " + s.checkBalance(str1));
       System.out.println("Expression 2: " + s.checkBalance(str2));
   }
}
```

Output:



```
ramyabharathi@Ramyas-Air day6(practise) % javac Sum13.java
ramyabharathi@Ramyas-Air day6(practise) % java Sum13
Expression 1: Balanced
Expression 2: Not Balanced
ramyabharathi@Ramyas-Air day6(practise) % 
```

Time Complexity:O(n)

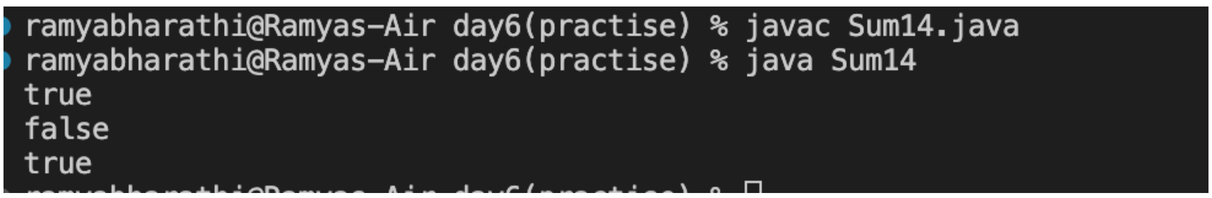
14.

Check if two Strings are Anagrams of each other
Given two strings s1 and s2 consisting of lowercase characters, the task is to check whether the two given strings are anagrams of each other or not. An anagram of a string is another string that contains the same characters, only the order of characters can be different.
Input: s1 = “geeks” s2 = “kseeg”
Output: true
Explanation: Both the string have same characters with same frequency. So, they are anagrams.
Input: s1 = “allergy” s2 = “allergic”
Output: false
Explanation: Characters in both the strings are not same. s1 has extra character „y‟ and s2 has extra characters „i‟ and „c‟, so they are not anagrams.
Input: s1 = “g”, s2 = “g”
Output: true
Explanation: Characters in both the strings are same, so they are anagrams.

Solution:

```java
class Sum14 {
   public boolean areAnagrams(String s1, String s2) {
       if (s1.length() != s2.length()) return false;
       int[] count = new int[26];
       for (int i = 0; i < s1.length(); i++) {
           count[s1.charAt(i) - 'a']++;
           count[s2.charAt(i) - 'a']--;
       }
       for (int c : count) {
           if (c != 0) return false;
       }
       return true;
   }
   public static void main(String[] args) {
       Sum14 s = new Sum14();
       String s1 = "geeks";
       String s2 = "kseeg";
       System.out.println(s.areAnagrams(s1, s2));

       String s3 = "allergy";
       String s4 = "allergic";
       System.out.println(s.areAnagrams(s3, s4));

       String s5 = "g";
       String s6 = "g";
       System.out.println(s.areAnagrams(s5, s6));
   }
}
```

Output:



```
ramyabharathi@Ramyas-Air day6(practise) % javac Sum14.java
ramyabharathi@Ramyas-Air day6(practise) % java Sum14
true
false
true
```

Time Complexity:O(n)

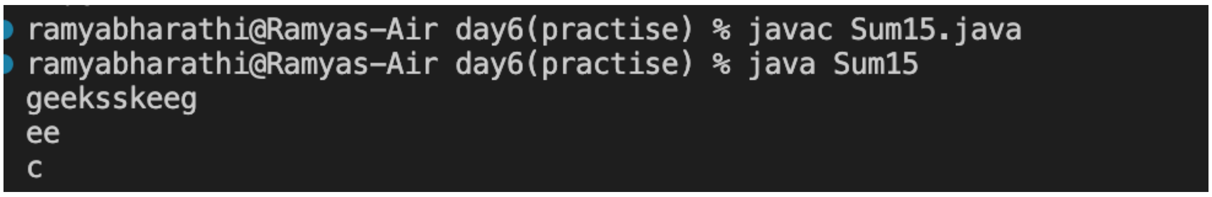
15.

Longest Palindromic Substring
Given a string str, the task is to find the longest substring which is a palindrome. If there are multiple answers, then return the first appearing substring.
Input: str = “forgeeksskeegfor”
Output: “geeksskeeg”
Explanation: There are several possible palindromic substrings like “kssk”, “ss”, “eeksskee” etc. But the substring “geeksskeeg” is the longest among all.
Input: str = “Geeks”
Output: “ee”
Input: str = “abc”
Output: “a”
Input: str = “”
Output: “”

Solution:

```java
class Sum15 {
   public String longestPalindrome(String s) {
       if (s == null || s.length() < 1) return "";
       int start = 0, end = 0;
       for (int i = 0; i < s.length(); i++) {
           int len1 = expandAroundCenter(s, i, i);
           int len2 = expandAroundCenter(s, i, i + 1);
           int len = Math.max(len1, len2);
           if (len > end - start) {
               start = i - (len - 1) / 2;
               end = i + len / 2;
           }
       }
       return s.substring(start, end + 1);
   }

   private int expandAroundCenter(String s, int left, int right) {
       while (left >= 0 && right < s.length() && s.charAt(left) == s.charAt(right)) {
           left--;
           right++;
       }
       return right - left - 1;
   }

   public static void main(String[] args) {
       Sum15 p = new Sum15();
       System.out.println(p.longestPalindrome("forgeeksskeegfor"));
       System.out.println(p.longestPalindrome("Geeks"));
       System.out.println(p.longestPalindrome("abc"));
       System.out.println(p.longestPalindrome(""));
   }
}
```

Output:



```
ramyabharathi@Ramyas-Air day6(practise) % javac Sum15.java
ramyabharathi@Ramyas-Air day6(practise) % java Sum15
geeksskeeg
ee
c
```

Time Complexity:O(n^2)

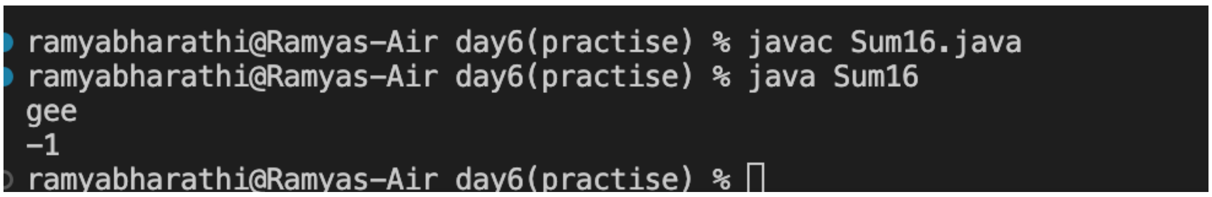
16.

Longest Common Prefix using Sorting
Given an array of strings arr[]. The task is to return the longest common prefix among each and every strings present in the array. If there"s no prefix common in all the strings, return "-1".
Input: arr[] = ["geeksforgeeks", "geeks", "geek", "geezer"]
Output: gee
Explanation: "gee" is the longest common prefix in all the given strings.Input: arr[] = ["hello", "world"]
Output: -1
Explanation: There"s no common prefix in the given strings.

Solution:

```
import java.util.Arrays;

class Sum16 {
   public String longestCommonPrefix(String[] arr) {
       if (arr == null || arr.length == 0) return "-1";
       Arrays.sort(arr);
       String first = arr[0];
       String last = arr[arr.length - 1];
       int i = 0;
       while (i < first.length() && i < last.length() && first.charAt(i) == last.charAt(i)) {
           i++;
       }
       return i == 0 ? "-1" : first.substring(0, i);
   }

   public static void main(String[] args) {
       Sum16 s = new Sum16();
       String[] arr1 = {"geeksforgeeks", "geeks", "geek", "geezer"};
       System.out.println(s.longestCommonPrefix(arr1));
       String[] arr2 = {"hello", "world"};
       System.out.println(s.longestCommonPrefix(arr2));
   }
}
```

Output:



```
ramyabharathi@Ramyas-Air day6(practise) % javac Sum16.java
ramyabharathi@Ramyas-Air day6(practise) % java Sum16
gee
-1
ramyabharathi@Ramyas-Air day6(practise) %
```

Time Complexity:O(nlogn)

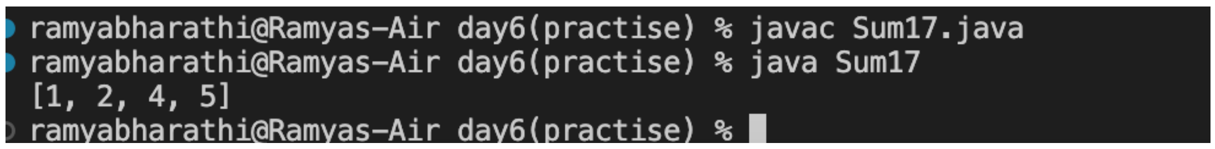
17.

Delete middle element of a stack
Given a stack with push(), pop(), and empty() operations, The task is to delete the middle element of it without using any additional data structure.
Input : Stack[] = [1, 2, 3, 4, 5]
Output : Stack[] = [1, 2, 4, 5]
Input : Stack[] = [1, 2, 3, 4, 5, 6]
Output : Stack[] = [1, 2, 4, 5, 6]

Solution:

```java
import java.util.Stack;

class Sum17 {
   static Stack<Integer> stack = new Stack<>();

   public static int count(Stack<Integer> stack) {
       if (stack.isEmpty()) return 0;
       int temp = stack.pop();
       int size = 1 + count(stack);
       stack.push(temp);
       return size;
   }

   public static void deleteMiddle(Stack<Integer> stack, int size, int currentIndex) {
       if (stack.isEmpty()) return;
       int temp = stack.pop();
       if (currentIndex == size / 2) {
           return;
       }
       deleteMiddle(stack, size, currentIndex + 1);
       stack.push(temp);
   }

   public static void main(String[] args) {
       stack.push(1);
       stack.push(2);
       stack.push(3);
       stack.push(4);
       stack.push(5);

       int size = count(stack);
       deleteMiddle(stack, size, 0);
       System.out.println(stack);
   }
}
```

Output:



```
ramyabharathi@Ramyas-Air day6(practise) % javac Sum17.java
ramyabharathi@Ramyas-Air day6(practise) % java Sum17
[1, 2, 4, 5]
ramyabharathi@Ramyas-Air day6(practise) %
```

Time Complexity:O(n)

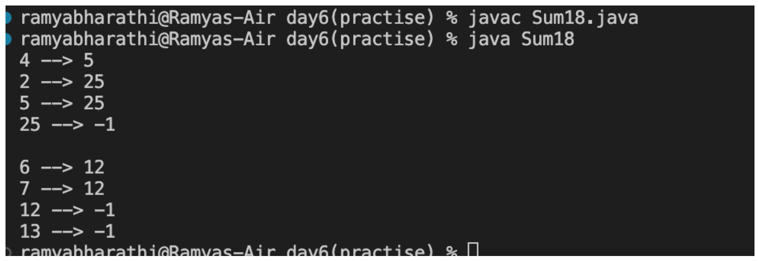
18.

Next Greater Element (NGE) for every element in given Array
Given an array, print the Next Greater Element (NGE) for every element.
Note: The Next greater Element for an element x is the first greater element on the right side of x in the array. Elements for which no greater element exist, consider the next greater element as -1.
Input: arr[] = [4 , 5 , 2 , 25]
Output: 4 –> 5
5 –> 25
2 –> 25
25 –> -1
Explanation: Except 25 every element has an element greater than them present on the right side
Input: arr[] = [13 , 7, 6 , 12]
Output: 13 –> -1
7 –> 12
6 –> 12
12 –> -1
Explanation: 13 and 12 don‟t have any element greater than them present on the right side

Solution:

```java
import java.util.Stack;

class Sum18 {
   public static void NGE(int[] arr) {
       Stack<Integer> s = new Stack<>();
       for (int i = 0; i < arr.length; i++) {
           while (!s.isEmpty() && arr[s.peek()] < arr[i]) {
               int idx = s.pop();
               System.out.println(arr[idx] + " --> " + arr[i]);
           }
           s.push(i);
       }
       while (!s.isEmpty()) {
           System.out.println(arr[s.pop()] + " --> -1");
       }
   }

   public static void main(String[] args) {
       int[] arr1 = { 4, 5, 2, 25 };
       int[] arr2 = { 13, 7, 6, 12 };
       NGE(arr1);
       System.out.println();
       NGE(arr2);
   }
}
```

Output:



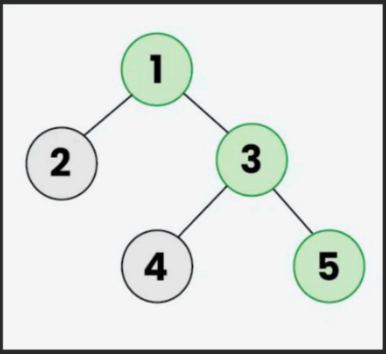
```
ramyabharathi@Ramyas-Air day6(practise) % javac Sum18.java
ramyabharathi@Ramyas-Air day6(practise) % java Sum18
4 --> 5
2 --> 25
5 --> 25
25 --> -1

6 --> 12
7 --> 12
12 --> -1
13 --> -1
ramyabharathi@Ramyas-Air day6(practise) %
```

Time complexity:O(n)

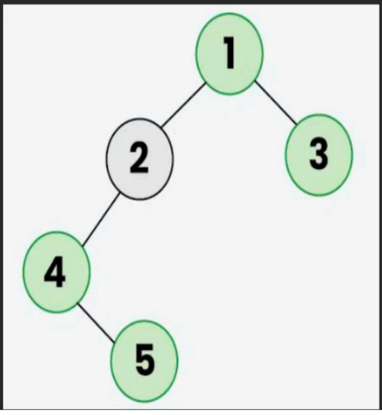
19.

Print Right View of a Binary Tree
Given a Binary Tree, the task is to print the Right view of it. The right view of a Binary Tree is a set of rightmost nodes for every level.

*Example 1: The **Green** colored nodes (1, 3, 5) represents the Right view in the below Binary tree.*



*Example 2: The **Green** colored nodes (1, 3, 4, 5) represents the Right view in the below Binary tree.*

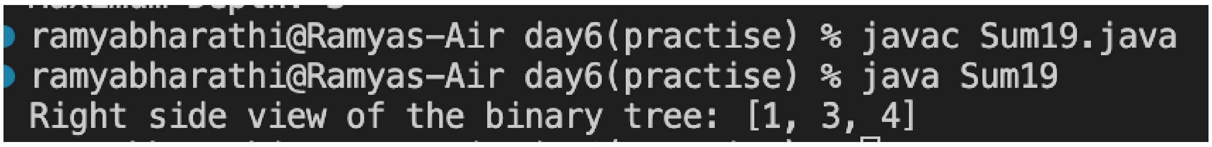


Solution:

```java
import java.util.*;
class TreeNode {
   int val;
   TreeNode left;
   TreeNode right;
   TreeNode(int val) { this.val = val; }
   TreeNode(int val, TreeNode left, TreeNode right) {
       this.val = val;
       this.left = left;
       this.right = right;
   }
}
public class Sum19 {
   public List<Integer> rightSideView(TreeNode root) {
       List<Integer> result = new ArrayList<>();
       if (root == null) return result;


       Queue<TreeNode> queue = new LinkedList<>();
       queue.offer(root);
```

```java
        while (!queue.isEmpty()) {
            int levelSize = queue.size();
            TreeNode rightMostNode = null;


            for (int i = 0; i < levelSize; i++) {
                TreeNode currentNode = queue.poll();
                rightMostNode = currentNode;
                if (currentNode.left != null) queue.offer(currentNode.left);
                if (currentNode.right != null) queue.offer(currentNode.right);
            }
            result.add(rightMostNode.val);
        }
        return result;
    }


    public static void main(String[] args) {
        TreeNode root = new TreeNode(1);
        root.left = new TreeNode(2);
        root.right = new TreeNode(3);
        root.left.right = new TreeNode(5);
        root.right.right = new TreeNode(4);


        Sum19 r = new Sum19();
        List<Integer> rightView = r.rightSideView(root);
        System.out.println("Right side view of the binary tree: " + rightView);
    }
}
```

Output:

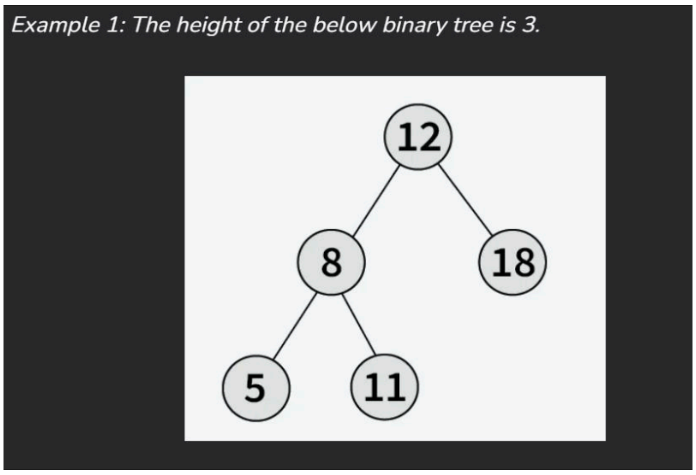


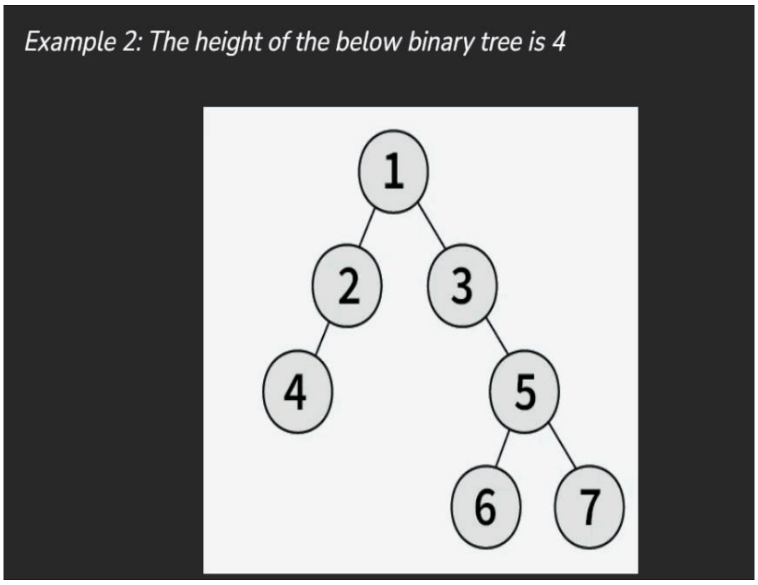
```
ramyabharathi@Ramyas-Air day6(practise) % javac Sum19.java
ramyabharathi@Ramyas-Air day6(practise) % java Sum19
Right side view of the binary tree: [1, 3, 4]
```

Time complexity:O(n)

20.

Maximum Depth or Height of Binary Tree
Given a binary tree, the task is to find the maximum depth or height of the tree. The height of the tree is the number of vertices in the tree from the root to the deepest node.

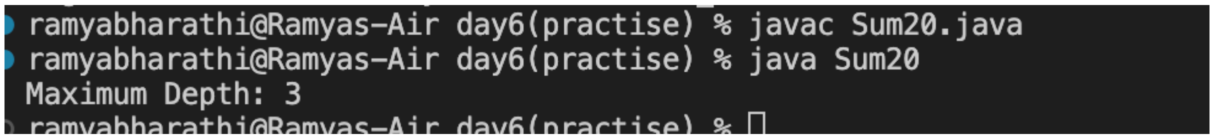




Solution:

```
class Node {
   int data;
   Node left, right;

   Node(int data) {
       this.data = data;
       left = right = null;
   }
}


class Sum20 {
   public int maxDepth(Node root) {
       if (root == null) return 0;
       int l = maxDepth(root.left);
       int r = maxDepth(root.right);
       return Math.max(l, r) + 1;
   }

   public static void main(String[] args) {
       Sum20 tree = new Sum20();
       Node root = new Node(12);
       root.left = new Node(8);
       root.right = new Node(18);
       root.left.left = new Node(5);
       root.left.right = new Node(11);

       System.out.println("Maximum Depth: " + tree.maxDepth(root));
   }
}
```

Output:



```
ramyabharathi@Ramyas-Air day6(practise) % javac Sum20.java
ramyabharathi@Ramyas-Air day6(practise) % java Sum20
Maximum Depth: 3
ramyabharathi@Ramyas-Air day6(practise) %
```

Time complexity:O(n)