**Day-9 DSA Practice**

**1)Valid palindrome**

Program:

public class ValidPalindrome {

public static void main(String[] args) {

ValidPalindrome obj=new ValidPalindrome();

String s="A man, a plan, a canal: Panama";

System.out.println(obj.valid(s));

}

boolean valid(String s){

//s=s.replaceAll("[^a-zA-Z0-9]","").toLowerCase();

s= s.toLowerCase();

StringBuffer str=new StringBuffer();

for(char ch:s.toCharArray()){

if(Character.isLetterOrDigit(ch)){

str.append(ch);

}

}

str.toString();

int left=0;

int right=str.length()-1;

while(left<right){

if(str.charAt(left)!=str.charAt(right)){

return false;

}

left++;

right--;

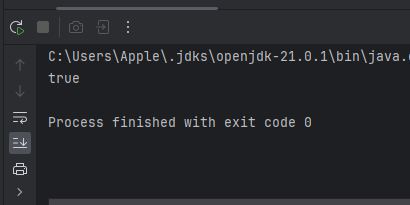
}

return true;

}

}

Output:



Timecomplexity:O(N)

SpaceComplexity:)O(N)

**2)Is Subsequence**

Program:

public class issubsequence {

public static void main(String[] args) {

issubsequence obj=new issubsequence();

String s = "abc", t = "ahbgdc";

System.out.println(obj.isSubsequence(s,t));

}

public boolean isSubsequence(String s, String t) {

int i=0,j=0;

while(i<s.length() && j<t.length()){

if(s.charAt(i)==t.charAt(j)){

i++;

}

j++;

}

if(i==s.length()){

return true;

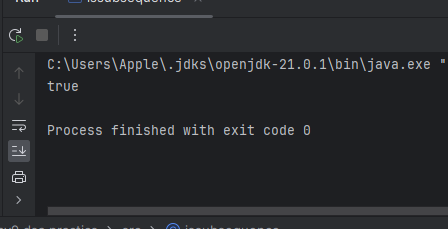
}

return false;

}

}

Output:



Timecomplexity:O(N)

SpaceComplexity:)O(1)

**3)Two Sum II**

Program:

import java.util.Arrays;

public class TwosumII {

public static void main(String[] args) {

TwosumII obj=new TwosumII();

int[] arr={2,7,11,15};

int target=9;

System.out.println(Arrays.toString(obj.twoSum(arr,target)));

}

public int[] twoSum(int[] numbers, int target) {

int start=0;

int end=numbers.length-1;

int[] arr=new int[2];

while(start<end){

int sum=numbers[start]+numbers[end];

if(sum==target){

arr[0]=start+1;

arr[1]=end+1;

}

if(sum<target){

start++;

}

else{

end--;

}

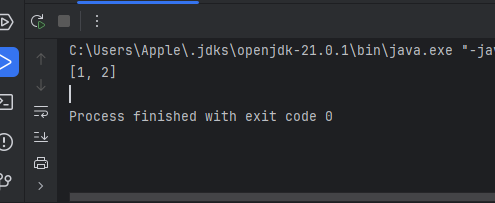
}

return arr;

}

}

Output:



Timecomplexity:O(N)

SpaceComplexity:)O(N)

**4)Container With Most Water**

Program:

public class ContainerwithmostWater {

public static void main(String[] args) {

ContainerwithmostWater obj=new ContainerwithmostWater();

int[] arr={1,8,6,2,5,4,8,3,7};

System.out.println(obj.check(arr));

}

int check(int[] arr){

int low=0;

int max=0;

int high=arr.length-1;

for(int i=0;i<arr.length;i++){

int height=Math.min(arr[low],arr[high]);

int width=high-low;

int area=height\*width;

max=Math.max(max,area);

if(arr[low]<arr[high]){

low++;

}

else{

high--;

}

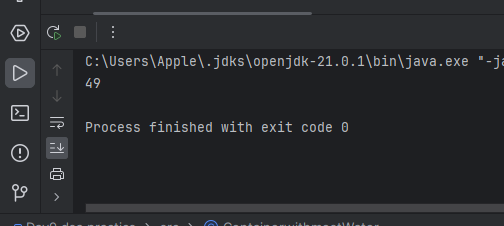
}

return max;

}

}

Output:



Timecomplexity:O(N)

SpaceComplexity:)O(1)

**5)3Sum**

Program:

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

public class threesum {

public static void main(String[] args) {

threesum obj=new threesum();

int[] nums = {-1, 0, 1, 2, -1, -4};

System.out.println(obj.threeSum(nums));

}

public List<List<Integer>> threeSum(int[] nums) {

Arrays.sort(nums);

List<List<Integer>> list = new ArrayList<>();

int n = nums.length;

for (int i = 0; i < n - 2; i++) {

if (i > 0 && nums[i] == nums[i - 1]) {

continue;

}

int start = i + 1;

int end = nums.length - 1;

while (start < end) {

int sum = nums[i] + nums[start] + nums[end];

if (sum == 0) {

list.add(Arrays.asList(nums[i], nums[start], nums[end]));

while (start < end && nums[start] == nums[start + 1]) {

start++;

}

while (start < end && nums[end] == nums[end - 1]) {

end--;

}

start++;

end--;

} else if (sum < 0) {

start++;

} else {

end--;

}

}

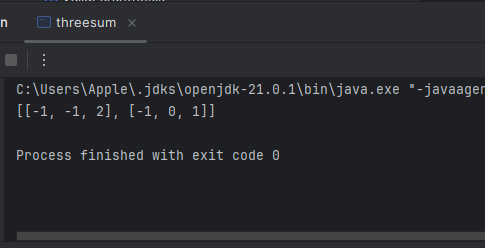
}

return list;

}

}

Output:



Timecomplexity:

SpaceComplexity:

**6)Minimum size subarraysum**

Program:

public class Minimumsizesubarraysum {

public static void main(String[] args) {

Minimumsizesubarraysum obj=new Minimumsizesubarraysum();

int[] arr={2,3,1,2,4,3};

int target=7;

System.out.println(obj.minSubArrayLen(target,arr));

}

public int minSubArrayLen(int target, int[] nums) {

int left=0;

int right=0;

int min=Integer.MAX\_VALUE;

int sum=0;

while(right<nums.length){

sum+=nums[right];

right++;

while(target<=sum){

min=Math.min(min,right-left);

sum-=nums[left];

left++;

}

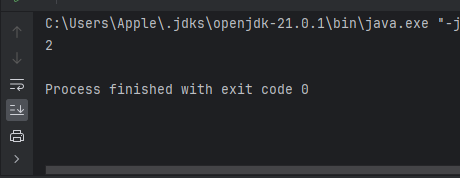
}

return min==Integer.MAX\_VALUE?0:min;

}

}

Output:



**7)Longest Substring withoutSubstring**

Program:

import java.util.HashMap;

public class LongestSubStringwithoutrepeatingCharacter {

public static void main(String[] args) {

LongestSubStringwithoutrepeatingCharacter obj=new LongestSubStringwithoutrepeatingCharacter();

String s="abcabcbb";

System.out.println(obj.lengthOfLongestSubstring(s));

}

public int lengthOfLongestSubstring(String s) {

int left=0;

int right=0;

int max=0;

HashMap<Character,Integer> map=new HashMap();

while(right<s.length()){

if(map.containsKey(s.charAt(right))){

left=Math.max(left,map.get(s.charAt(right))+1);

}

map.put(s.charAt(right),right);

max=Math.max(max,right-left+1);

right++;

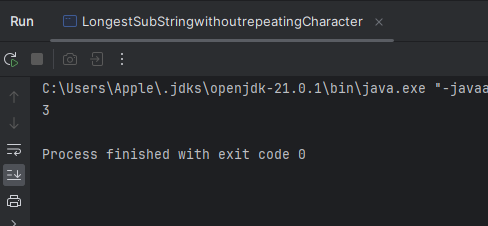
}

return max;

}

}

Output:



**10)Valid Paranthesis**

Program:

import java.util.Stack;

public class ValidParanthesis {

public static void main(String[] args) {

ValidParanthesis obj=new ValidParanthesis();

String s="()[]{}";

System.out.println(obj.isValid(s));

}

public boolean isValid(String s) {

Stack<Character> stack=new Stack<>();

for(int i=0;i<s.length();i++){

char c=s.charAt(i);

if(c=='(' ||c=='{' ||c=='['){

stack.push(c);

}else{

if(stack.isEmpty()){

return false;

}

char top=stack.pop();

if((c==')'&&top!='(')||

(c=='}' && top!='{')||

(c==']' && top!='[')

){

return false;

}

}

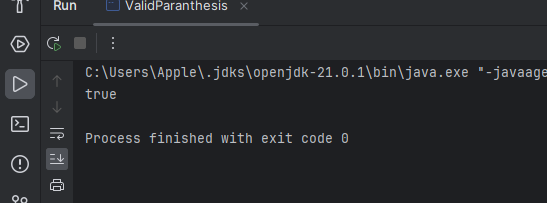
}

return stack.isEmpty();

}

}

Output:



**11)Simplify Path**

Program:

import java.util.Stack;

public class Simplifypath {

public static void main(String[] args) {

Simplifypath obj=new Simplifypath();

String s="/.../a/../b/c/../d/./";

System.out.println(obj.simplifyPath(s));

}

public String simplifyPath(String path) {

String[] str=path.split("/");

Stack<String> stack=new Stack<>();

for(int i=0;i<str.length;i++){

if(str[i].equals("..")){

if(!stack.isEmpty()){

stack.pop();

}

}

else if(!str[i].equals("") && !str[i].equals(".")){

stack.push(str[i]);

}

}

StringBuilder str1=new StringBuilder();

while(!stack.isEmpty()){

str1.insert(0,"/"+stack.pop());

}

if(str1.length()==0){

return "/";

}

else{

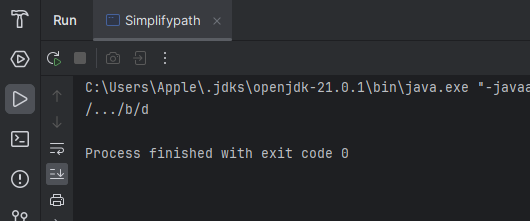
return str1.toString();

}

}

}

Output:



**14)Evaluate Reverse Polish**

Program:

import java.util.Stack;

public class Reversepolisgnotatio {

public static void main(String[] args) {

Reversepolisgnotatio obj=new Reversepolisgnotatio();

String[] str={"2","1","+","3","\*"};

System.out.println(obj.evalRPN(str));

}

public int evalRPN(String[] tokens) {

Stack<Integer> stack = new Stack<>();

for (String str : tokens) {

if (str.equals("+") || str.equals("-") || str.equals("\*") || str.equals("/")) {

int b = stack.pop();

int a = stack.pop();

int res = 0;

switch (str) {

case "+":

res = a + b;

break;

case "-":

res = a - b;

break;

case "\*":

res = a \* b;

break;

case "/":

res = a / b;

break;

}

stack.push(res);

} else {

stack.push(Integer.parseInt(str));

}

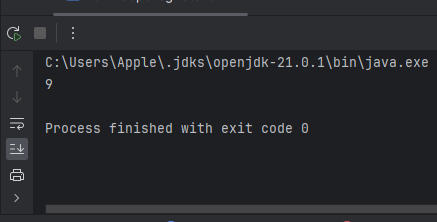
}

return stack.pop();

}

}

Output:



**16)Search Insert Position**

Program:

public class SearchInsearchposition {

public static void main(String[] args) {

SearchInsearchposition obj=new SearchInsearchposition();

int[] arr={1,3,5,6};

int target=5;

System.out.println(obj.search(arr,target));

}

int search(int[] arr,int target){

int start=0;

int end=arr.length-1;

while(start<end){

int mid=start+(end-start)/2;

if(arr[mid]==target){

return mid;

}

else if(arr[mid]>target){

end=end-1;

}

else{

start=start+1;

}

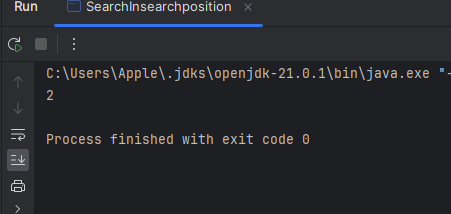
}

return start;

}

}

Output:



**17)Search a 2D Matrix:**

Program:

public class search2dMatrix {

public static void main(String[] args) {

search2dMatrix obj=new search2dMatrix();

int[][] matrix={{1,3,5,7},{10,11,16,20},{23,30,34,60}};

int target=3;

System.out.println(obj.search(matrix,target));

}

boolean search(int[][] matrix,int target){

int row=0;

int col=matrix[0].length-1;

while(row<col){

if(matrix[row][col]==target){

return true;

}

else if(matrix[row][col]<target){

row++;

}

else{

col--;

}

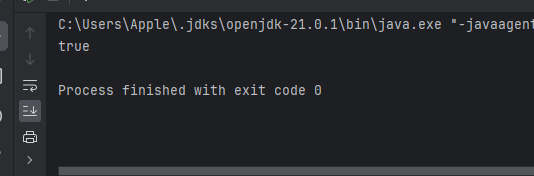
}

return false;

}

}

Output:



**18)Find Peak Element:**

Program:

public class findpeekelement {

public static void main(String[] args) {

findpeekelement obj=new findpeekelement();

int[] arr={1,2,3,1};

System.out.println(obj.fint(arr));

}

int fint(int[] arr){

int low=0;

int high=arr.length-1;

while(low<high){

int mid=low+(high-low)/2;

if(arr[mid]<arr[mid+1]){

low=mid+1;

}

else{

high=mid;

}

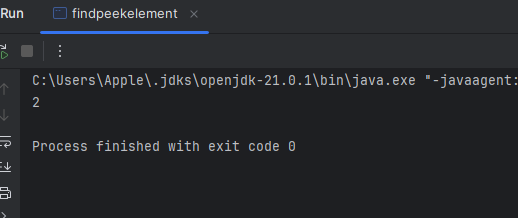
}

return low;

}

}

Output:



19)Search in Roated sorted array:

Program:

public class SearchinrotatedSortedarray {

public static void main(String[] args) {

SearchinrotatedSortedarray obj=new SearchinrotatedSortedarray();

int[] arr={4,5,6,7,0,1,2};

int target=0;

System.out.println(obj.search(arr,target));

}

int search(int[] arr,int target){

int low=0;

int high=arr.length-1;

while(low<=high){

int mid=low+(high-low)/2;

if(arr[mid]==target){

return mid;

}

if(arr[mid]>=arr[low]){

if(arr[mid]>target && target>=arr[low]){

high=mid-1;

}else{

low=mid+1;

}

}else{

if(arr[mid]<target && target<=arr[high]){

low=mid+1;

}else{

high=mid-1;

}

}

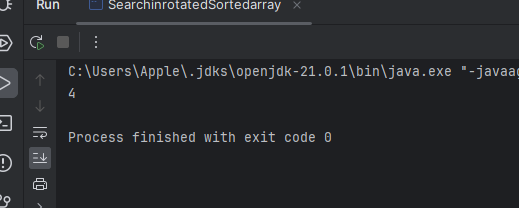
}

return -1;

}

}

Output:



**20)First and Last Position**

Program:

import java.util.ArrayList;

public class FirstandLastOccurances {

public static void main(String[] args) {

FirstandLastOccurances obj= new FirstandLastOccurances();

int[] arr={1, 3, 5, 5, 5, 5, 67, 123, 125};

int x=5;

System.out.println(obj.find(arr,x));

}

ArrayList<Integer> find(int arr[], int x) {

// code here

int n=arr.length;

ArrayList<Integer> li=new ArrayList<>();

li.add(first(arr,0,n-1,x));

li.add(last(arr,0,n-1,x));

return li;

}

int first(int[] arr,int start,int end,int x){

int res=-1;

while(start<=end){

int mid=start+(end-start)/2;

if(arr[mid]>x){

end=mid-1;

}

else if(arr[mid]<x){

start=mid+1;

}

else{

res=mid;

end=mid-1;

}

}

return res;

}

int last(int[] arr,int start,int end,int x){

int res=-1;

while(start<=end){

int mid=start+(end-start)/2;

if(arr[mid]>x){

end=mid-1;

}

else if(arr[mid]<x){

start=mid+1;

}

else{

res=mid;

start=mid+1;

}

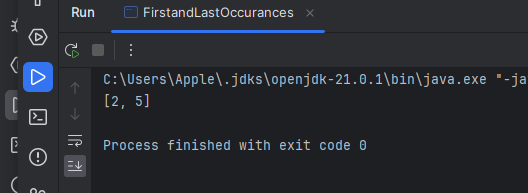
}

return res;

}

}

Output:



**21)Find Minimum in Rotated sortedArray**

Program:

public class Findminimuminsortedarray {

public static void main(String[] args) {

Findminimuminsortedarray obj=new Findminimuminsortedarray();

int[] arr={3,4,5,1,2};

System.out.println(obj.search(arr));

}

int search(int[] arr){

int start=0;

int end=arr.length-1;

while(start<end){

int mid=start+(end-start)/2;

if(arr[end]<arr[mid]){

start=mid+1;

}else{

end=mid;

}

}

return arr[start];

}

}

Output:

