**C-DAC Mumbai Date 26/09/2024**

**Subject: Algorithm and Data Structure**

**Assignment 2**

**Solve the assignment with following thing to be added in each question.**

-Program

-Flow chart

-Explanation

-Output

-Time and Space complexity

1. Printing Patterns

Problem: Write a Java program to print patterns such as a right triangle of stars.

Solution--

**package** assign2;

**import** java.util.Scanner;

**public** **class** A2\_Q1 {

**public** **static** **void** main(String args[]) {

Scanner sc=**new** Scanner(System.***in***);

System.***out***.print("n = ");

**int** n=sc.nextInt();

**for**(**int** i=1; i<=n;i++) {

**for**(**int** j=1;j<=i;j++) {

System.***out***.print("\*");

}

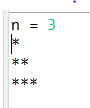
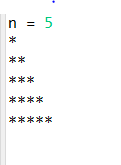
System.***out***.println();

}

sc.close();

}

}



Time Complexity—O(n2)

Space complexity—O(1)

2. Remove Array Duplicates

Problem: Write a Java program to remove duplicates from a sorted array and return the new length of the array.

**Solution—**

**package** assign2;

**import** java.util.Scanner;

**public** **class** A2\_Q2 {

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

System.***out***.println("Enter no of elements:");

**int** size=sc.nextInt();

**int** arr[]=**new** **int**[size];

**for**(**int** i=0;i<size;i++)

arr[i]=sc.nextInt();

**int** idx = 1; // Start from the second element

**for** (**int** i = 1; i < size; i++) {

**if** (arr[i] != arr[i - 1]) {

arr[idx++] = arr[i];

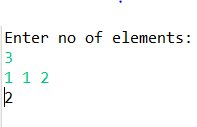
}

}

System.***out***.print(idx);

}

}



Time complexity—O(n)

Space complexity—O(n)

3. Remove White Spaces from String

Problem: Write a Java program to remove all white spaces from a given string.

**package** assign2;

**import** java.util.\*;

**public** **class** A2\_Q3 {

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

String s=sc.nextLine();

String[] words=s.split(" ");

**for**(**int** i=0;i<words.length;i++)

System.***out***.print(words[i]);

sc.close();

}

}

Test Cases:

Input: "Hello World"

Output: "HelloWorld"

Input: " Java Programming "

Output: "JavaProgramming"

4. Reverse a String

Problem: Write a Java program to reverse a given string.

Solution—

**package** com.stack;

**class** S1tack{

**private** **int** max;

**private** **int** top;

**private** **int** a[];

S1tack(**int** n){

top=-1;

max=n;

a=**new** **int**[n];

}

**boolean** isEmpty()

{

**return** top==-1?**true**:**false**;

}

**boolean** isFull() {

**return** (top>max-1);

}

**public** **void** push(**int** x) {

**if** (!isFull()) {

a[++top] = x; // Increment top and add element

} **else** {

System.***out***.println("Stack is full. Push operation failed.");

}

}

**int** pop() {

**if**(isEmpty()) {

System.***out***.println("Stack underflow");

**return** -1;

}

**return** a[top--];

}

**int** peek() {

**if**(!isEmpty()) {

**return** a[top];

}

**else**

System.***out***.println("Stack is empty");

**return** -1;

}

**void** display() {

**if**(isEmpty()) {

System.***out***.println("Stack is empty");

}

**else**

**for**(**int** i:a) {

System.***out***.println(i);

}

}

}

**public** **class** StackApp {

**public** **static** **void** reverse(StringBuffer str) {

**int** n = str.length();

S1tack s1 = **new** S1tack(n); // Create a stack of the same size as the string

// Push each character of the string into the stack

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

s1.push(str.charAt(i));

}

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

**char** ch=(**char**)s1.pop();

str.setCharAt(i, ch);

}

}

**public** **static** **void** main(String[] args) {

StringBuffer s = **new** StringBuffer("CDAC Mumbai");

*reverse*(s); // Reverse the string

System.***out***.println("Reverse of the string = " + s);

}

}

Time complexity— O(n)

Space Complexity—O(n)

5. Reverse Array in Place

Problem: Write a Java program to reverse an array in place.

**package** assign2;

**import** java.util.Arrays;

**import** java.util.Scanner;

**public** **class** A2\_Q5 {

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

System.***out***.println("enter no of elements:");

**int** n=sc.nextInt();

**int** a[]=**new** **int**[n];

**int** b[]=**new** **int**[n];

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

a[i]=sc.nextInt();

}

**int** j=n-1;

**for**(**int** i=j;i>=0;i--)

{

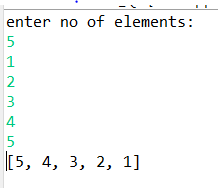
b[j-i]=a[i];

}

System.***out***.println(Arrays.*toString*(b));

}

}



Time Complexity= O(n)

Space complexity= O(n)

6. Reverse Words in a String

Problem: Write a Java program to reverse the words in a given sentence.

**package** assign2;

**import** java.util.Scanner;

**public** **class** A2\_Q6 {

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

String s=sc.nextLine();

String[] words=s.split(" ");

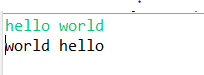
String temp=words[0];

words[0]=words[1];

words[1]=temp;

System.***out***.println(String.*join*(" ", words));

}

Time complexity—O(n)

Space complexity—O(n)

7. Reverse a Number

Problem: Write a Java program to reverse a given number.

**package** assign2;

**import** java.util.Scanner;

**public** **class** A2\_Q7 {

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

**int** n=sc.nextInt();

**int** sign=n>0?1:-1;

n=Math.*abs*(n);

**int** n1=0;

**while**(n>0) {

**int** rem=n%10;

n1=n1\*10+rem;

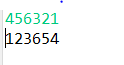
n=n/10;

}

System.***out***.println(n1\*sign);

}

}

Time complexity—O(n)

Space complexity—O(1)

8. Array Manipulation

Problem: Perform a series of operations to manipulate an array based on range update queries. Each query adds a value to a range of indices.

**package** assign2;

**import** java.util.\*;

**public** **class** A2\_Q8{

**public** **static** **void** main(String... args){

Scanner sc = **new** Scanner(System.***in***);

**int** size = sc.nextInt();

**int** operations = sc.nextInt();

**int** maxNum = 0;

**int**[] arr = **new** **int**[size];

**for**(**int** i=0; i<operations; i++){

**int** idx1 = sc.nextInt()-1;//0

**int** idx2 = sc.nextInt()-1;//1

**int** num = sc.nextInt();

**for**(**int** j=idx1; j<=idx2; j++){

arr[j] += num;

maxNum = arr[j]>maxNum ? arr[j] : maxNum;

}

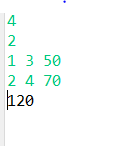
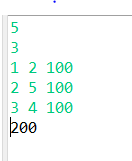
}

System.***out***.println(maxNum);

sc.close();

}

}

Test Cases:

Input: n = 5, queries = [[1, 2, 100], [2, 5, 100], [3, 4, 100]]

Output: 200

Input: n = 4, queries = [[1, 3, 50], [2, 4, 70]]

Output: 120

9. String Palindrome

Problem: Write a Java program to check if a given string is a palindrome.

**package** assign2;

**import** java.util.Scanner;

**public** **class** A2\_Q9 {

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

String str=sc.next();

**int** n=str.length();

**char**[] c=**new** **char**[n];

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

n--;

c[i]=str.charAt(n);

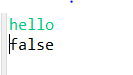
}

String c1= String.*valueOf*(c);

System.***out***.println(c1.equals(str));

}

}

Time complexity— O(n)

Space complexity=O(n)

10. Array Left Rotation

Problem: Write a Java program to rotate an array to the left by d positions.

**package** assign2;

**import** java.util.Arrays;

**import** java.util.Scanner;

**public** **class** A2\_Q10 {

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

System.***out***.println("enter no of elements in array:");

**int** n=sc.nextInt();

**int** a[]= **new** **int**[n];

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

a[i]=sc.nextInt();

}

System.***out***.println("enter no of positions to rotate:");

**int** d= sc.nextInt();

**int** a1[]=**new** **int**[d];

**for**(**int** i=0;i<d;i++) {

a1[i]=a[i];

}

**int** j=0;

**for**(**int** i=d; i<a.length;i++ )

{

a[j]=a[i];

j++;

}

**int** k=0;

**for**(**int** i=n-d;i<n;i++) {

a[i]=a1[k];

k++;

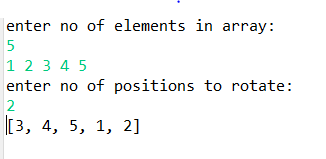
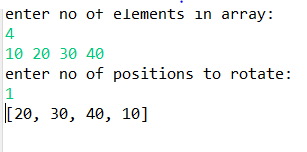
}

System.***out***.print(Arrays.*toString*(a));

sc.close();

}

}

Time complexity—O(n)

Space complexity—O(n)

Test Cases:

Input: arr = [1, 2, 3, 4, 5], d = 2

Output: [3, 4, 5, 1, 2]

Input: arr = [10, 20, 30, 40], d = 1

Output: [20, 30, 40, 10]