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

**Subject: Algorithm and Data Structure**

**Assignment 1**

**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.**

**Test Cases:**

Input: n = 3

Output:

\*

\*\*

\*\*\*

Input: n = 5

Output:

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

**Algorithm:**

Step:1 Input the number of rows

Step:2. Understand the Pattern

Step:3. Initialize the Loop for Rows

Step 4. Nested Loop for Printing Stars

Step 5. Print pattern.

**Program:**

public class Patterns{

public static void printpattern(int n){

int i;

int j;

for(i = 0 ; i <= n; i++)

{

for(j = 0; j <= i; j++)

{

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

}

System.out.println();

}

}

public static void main(String args[])

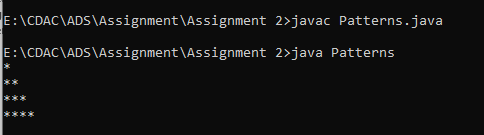
{

printpattern(3);

}

}

**Output:**



Time complexity: O(n)

**2. Remove Array Duplicates**

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

**Test Cases:**

Input: arr = [1, 1, 2]

Output: 2

Input: arr = [0, 0, 1, 1, 2, 2, 3, 3]

Output: 4

1.The array is already sorted,

2. Initialize two pointer i and index

3. we increment i to skip the duplicate.

4. When we reach nums[i] != nums[i] the duplicate run has ended so we must copy its value to nums[i + 1]nums[i+1].

5.i is then incremented and we repeat the same process again until index reaches the end of array.

**Program:**

public class ArrayDuplicate{

public int RemoveDup(int arr[])

{

int index=1;

for(int i=0; i < arr.length-1 ;i++)

{

if (arr[i]!=arr[i+1])

{

arr[index]=arr[i+1];

index++;

}

else{

continue;

}

}

return index;

}

public static void main(String[] args)

{

ArrayDuplicate a= new ArrayDuplicate();

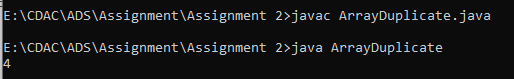
int[] arr={0, 0, 1, 1, 2, 2, 3, 3};

System.out.println(a.RemoveDup(arr));

}

}

Output:



**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.**

**Test Cases:**

Input: "Hello World"

Output: "HelloWorld"

Input: " Java Programming "

Output: "JavaProgramming"

1.Convert string to a character array.

2.Declare a temporary string.

3.Traverse the character array.

4.Check if the present character is white space or not.

5.If it is not white space, then add it to a temporary string.

6.Copy temporary string to original string.

7.Print the string.

**Program:**

public class SpaceArray{

public static void main(String[] args)

{ String s="Hello World";

char[] arr= s.toCharArray();

String temp="";

for(char c:arr){

if(c!=' ')

{

temp +=c;

}

}

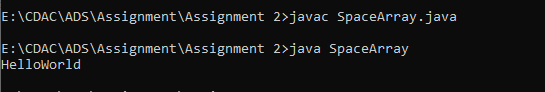
s=temp;

System.out.println(s);

}

}

**Output:**



**Time Complexity:** O(n)

**Space Complexity:** O(n)

**4. Reverse a String**

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

**Test Cases:**

Input: "hello"

Output: "olleh"

Input: "Java"

Output: "avaJ"

**Explanation:**

class Stringsrev {

public static void main (String[] args) {

String str= "hello",

String newstr=" ";

char ch;

System.out.println("hello");

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

{

ch= str.charAt(i);

newstr= ch+newstr;

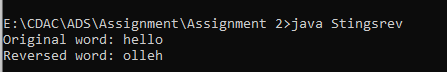
}

System.out.println("Reversed word: "+ newstr);

}

}

Output:



**Time Complexity:** O(n)

**Space Complexity:** O(n)

**5. Reverse Array in Place**

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

Test Cases:

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

Output: [4, 3, 2, 1]

Input: arr = [7, 8, 9]

Output: [9, 8, 7]

**Explaination:**

STEP 1: start

STEP 2: Initialize array {1, 2, 3, 4, 5}

STEP 3: print "Original Array:"

STEP 4: repeat step 5 for(i=0; i<arr.length ; i++)

STEP 5: print array

STEP 6: print statemet "Array in reverse order"

STEP 7: repeat step 8 for(i= arr.length-1; i>=0; i--)

STEP 8: print a[i]

STEP 9: end

**Program:**

class Arrayrev{

public static void main(String[] args){

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

System.out.println("Original Array");

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

{

System.out.print(arr[i]+"");

}

System.out.println(" ");

System.out.println("Array in reverse order");

for(int i=arr.length-1;i>=0;i--)

{

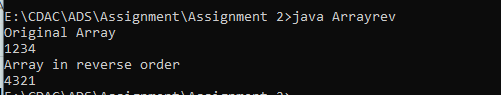
System.out.print(arr[i]+"");

}

}

}

Output:



**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.**

**Test Cases:**

Input: "Hello World"

Output: "World Hello"

Input: "Java Programming"

Output: "Programming Java"

**Explaination:**

1.check the string from the last character, and move towards the first character.

2.While check, if a space character is encountered, put a ‘ ‘ in that position and print the remaining string just after the ‘ ‘ character.

3.Repeat this until the loop is over and when the loop ends, print the string, the %s will make the printing of characters until it encounters the first ‘ ‘ character.

**Program:**

import java.util.\*;

class StringWordRev{

public static String revString(String str)

{

int i=str.length() - 1;

int start,end = i + 1;

String result = "";

while(i >= 0){

if(str.charAt(i) == ' '){

start = i+1;

while(start!=end)

result += str.charAt(start++);

result += " ";

end = i;

}

i--;

}

start = 0;

while (start != end)

result += str.charAt(start++);

return result;

}

public static void main(String[] args) {

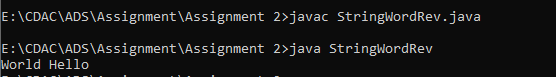
String str = "Hello World";

System.out.print(revString(str));

}

}

**Output:**



**7. Reverse a Number**

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

Test Cases:

Input: 12345

Output: 54321

Input: -9876

Output: -6789

**Explaination:**

Step1.First, we find the remainder of the given number by using the modulo (%) operator.

Step 2.Multiply the variable reverse by 10 and add the remainder into it.

Step3. Divide the number by 10.

**Program:**

class RevNumber{

public static void main(String[] args)

{

int n=-1234;

int rev=0;

while(n!=0)

{

int rem=n%10;

rev = rev\*10+rem;

n=n/10;

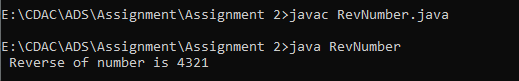
}

System.out.println(" Reverse of number is " +rev);

}

}

**Output:**



**Space Complexity:**O(log(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.**

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

Explaination:

1 Create an array arr of size n + 1 with zeros.

2.The extra element is used to handle boundary cases.

3. Iterate through each query [start, end, value].

4. Add value to arr[start].

5.Subtract value from arr[end + 1].

6.This is a prefix sum array update technique.

7.Calculate Prefix Sum:

8.Iterate through arr from index 1 to n.

9. Add the value at the previous index to the current index.

10. This calculates the cumulative sum.

11. Find Maximum Value:

12 Iterate through arr from index 1 to n.

13 Keep track of the maximum value.

**Program:**

import java.util.\*;

class ArrayManipulation {

public static void arrayManipulation(int n, int[][] queries) {

long[] arr = new long[n + 1];

for (int[] query : queries) {

int start = query[0];

int end = query[1];

int value = query[2];

arr[start] += value;

if (end + 1 <= n) {

arr[end + 1] -= value;

}

}

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

arr[i] += arr[i - 1];

}

long max = 0;

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

max = Math.max(max, arr[i]);

}

System.out.println(max);

}

public static void main(String[] args) {

int n = 5;

int[][] queries = {{1, 2, 100}, {2, 5, 100}, {3, 4, 100}};

arrayManipulation(n, queries);

}

}

Output:

Time complexity: O(n )

Space complexity: O(n)

**9. String Palindrome**

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

Test Cases:

Input: "madam"

Output: true

Input: "hello"

Output: false

**Explaination**:

1. for loop to reverse the string

-The loop runs from the end to the beginning of the string.

-The charAt() method accesses each character of the string.

-Each character of the string is accessed in reverse order and stored in reverseStr.

2. if statement to compare str and reverseStr

The toLowerCase() method converts both str and reverseStr to lowercase. This is because Java is case sensitive and 'm' and 'M' are two different values.

The equals() method checks if two strings are equal.

**Program:**

class Palindrome {

public static void main(String[] args) {

String str = "madam", reverseStr = "";

int strLength = str.length();

for (int i = (strLength - 1); i >=0; --i) {

reverseStr = reverseStr + str.charAt(i);

}

if (str.toLowerCase().equals(reverseStr.toLowerCase())) {

System.out.println(str + " True");

}

else {

System.out.println(str + " False");

}

}

}

**Output:**



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.**

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]

**Explaination:**

STEP 1: START

STEP 2: Initialize array {1, 2, 3, 4, 5 }.

STEP 3: set n =3

STEP 4: print "Original Array"

STEP 5: repeat step 6 for each element of array to print

STEP 6: print arr[i]

STEP 7: repeat step 8 to step 12 until the set

STEP 8: define j, first.

STEP 9: first = arr[0]

STEP 10: repeat step 11 for(j= 0; j<arr.length-1; j++)

STEP 11: arr[j]= arr[j+1]

STEP 12: arr[j]= first

STEP 13: print "Array after left rotation"

STEP 14: Repeat step 15 for each element of array to print

STEP 15: PRINT arr[i]

STEP 16: END

**Program:**

class RotateLeft {

public static void main(String[] args) {

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

int n = 2;

System.out.println("Original array: ");

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

System.out.print(arr[i] + " ");

}

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

int j, first;

first = arr[0];

for(j = 0; j < arr.length-1; j++){

arr[j] = arr[j+1];

}

arr[j] = first;

}

System.out.println();

System.out.println("Array after left rotation: ");

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

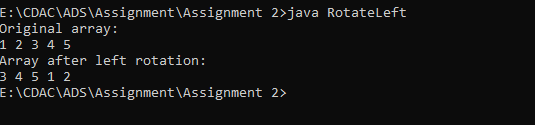
System.out.print(arr[i] + " ");

}

}

}

**Output:**



Time complexity:O(n^2)

Space complexity:O(1)