**Assignment 2**

**Name:** Yash Bandu Dhole

**Center:** Juhu

1. Printing Patterns

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

**Code:**

package problem1;

import java.util.Scanner;

public class pattern {

public static void main(String[] args) {

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

System.***out***.println("Enter number of rows: ");

int n = sc.nextInt();

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

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

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

}

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

}

}

}

**O/p:**

Enter number of rows:

5

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

**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.

**Code:**

package problem2;

public class removeDuplicates {

public static int rmdupli(int[] arr) {

if (arr.length==0) return 0;

int newLength = 1;

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

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

arr[newLength] = arr[i];

newLength++;

}

}

return newLength;

}

public static void main(String[] args) {

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

int length = *rmdupli*(arr);

System.***out***.println(" New Length " + length);

}

}

**O/p:**

New Length 4

**Time complexity:** O(n)

**Space complexity;** O(1)

3. Remove White Spaces from String

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

**Code:**

package problem3;

public class removeWhiteSpaces {

public static String removeSpaces(String str){

StringBuilder result = new StringBuilder();

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

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

result.append(str.charAt(i));

}

}

return result.toString();

}

public static void main(String[] args) {

String input = " Java Programming ";

String output = *removeSpaces*(input);

System.***out***.println(" Output: "+ output);

}

}

**O/p:**

Output: JavaProgramming

**Time complexity:** O(n)

**Space complexity:** O(n)

**Flowchart:**

4. Reverse a String

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

**Code:**

package problem4;

public class reverseString {

public static String reverse(String str) {

StringBuilder reversed = new StringBuilder();

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

reversed.append(str.charAt(i));

}

return reversed.toString();

}

public static void main(String[] args) {

String input = "hello";

String output = *reverse*(input);

System.***out***.println("Reversed: " + output);

}

}

**O/p:**

Reversed: olleh

**Time complexity:** O(n)

**Space complexity:** O(n)

5. Reverse Array in Place

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

**Code:**

package problem5;

public class reverseArray {

public static void reverse (int[] arr) {

int left = 0;

int right = arr.length -1;

while (left < right) {

int temp = arr[left];

arr[left]= arr[right];

arr[right] = temp;

left++;

right--;

}

}

public static void main(String[] args) {

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

*reverse*(arr);

System.***out***.println("Reversed Array: ");

for(int num: arr) {

System.***out***.println(num + " ");

}

}

}

**O/p:**

Reversed Array:

4

3

2

1

**Time complexity:**  O(n)

**Space complexity:** O(1)

6. Reverse Words in a String

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

**Code:**

package problem6;

public class reverseWord {

public static String revWords(String sentence) {

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

StringBuilder reversed = new StringBuilder();

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

reversed.append(words[i]);

if (i != 0) {

reversed.append(" ");

}

}

return reversed.toString();

}

public static void main(String[] args) {

String input = "Hello World";

String output = *revWords*(input);

System.***out***.println("Reversed Sentence: " + output);

}

}

**O/p:**

Reversed Sentence: World Hello

**Time complexity:** O(n)

**Space complexity:** O(n)

7. Reverse a Number

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

**Code:**

package problem7;

public class reverseNumber {

public static int reverse(int num) {

int reversed = 0;

int sign = (num < 0) ? -1 : 1;

num = Math.*abs*(num);

while (num != 0) {

int digit = num % 10;

reversed = reversed \* 10 + digit;

num /= 10;

}

return reversed \* sign;

}

public static void main(String[] args) {

int input = 12345;

int output = *reverse*(input);

System.***out***.println("Reversed Number: " + output);

}

}

**O/p:**

Reversed Number: 54321

**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.

**Code:**

package problem8;

public class RangeUpdate {

public static long getFinalSum(int n, int[][] queries) {

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

for (int[] query : queries) {

int start = query[0];

int end = query[1];

long value = query[2];

arr[start] += value;

if (end + 1 <= n) {

arr[end + 1] -= value;

}

}

long sum = 0;

long current = 0;

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

current += arr[i];

sum += current;

}

return sum;

}

public static void main(String[] args) {

int n1 = 5;

int[][] queries1 = {{1, 2, 100}, {2, 5, 100}, {3, 4, 100}};

System.***out***.println("Output: " + *getFinalSum*(n1, queries1));

int n2 = 4;

int[][] queries2 = {{1, 3, 50}, {2, 4, 70}};

System.***out***.println("Output: " + *getFinalSum*(n2, queries2));

}

}

**O/p:**

Output: 800

Output: 360

**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.

**Code:**

package problem9;

public class stringPalindrom {

public static boolean isPalindrome(String str) {

int left = 0;

int right = str.length() - 1;

while (left < right) {

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

return false;

}

left++;

right--;

}

return true;

}

public static void main(String[] args) {

String input = "radar";

boolean result = *isPalindrome*(input);

System.***out***.println("Is \"" + input + "\" a palindrome? " + result);

}

}

**O/p:**

Is "radar" a palindrome? true

**Time complexity:** O(n)

**Space complexity:** O(1)

10. Array Left Rotation

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

**Code:**

package problem10;

public class ArrayRotation {

public static void rotateLeft(int[] arr, int d) {

int n = arr.length;

d = d % n;

*reverseArray*(arr, 0, d - 1);

*reverseArray*(arr, d, n - 1);

*reverseArray*(arr, 0, n - 1);

}

public static void reverseArray(int[] arr, int start, int end) {

while (start < end) {

int temp = arr[start];

arr[start] = arr[end];

arr[end] = temp;

start++;

end--;

}

}

public static void main(String[] args) {

int[] arr1 = {1, 2, 3, 4, 5};

int d1 = 2;

*rotateLeft*(arr1, d1);

for (int num : arr1) {

System.***out***.print(num + " ");

}

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

int[] arr2 = {10, 20, 30, 40};

int d2 = 1;

*rotateLeft*(arr2, d2);

for (int num : arr2) {

System.***out***.print(num + " ");

}

}

}

**O/p:**

3 4 5 1 2

20 30 40 10

**Time complexity:** O(n)

**Space complexity:** O(1)