Assignment 2

1. **Printing Patterns**

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

import java.util.Scanner;

public class StarPattern{

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

System.out.print("Enter a number: ");

int n = sc.nextInt();

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

for(int col=1; col<=row; col++){

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

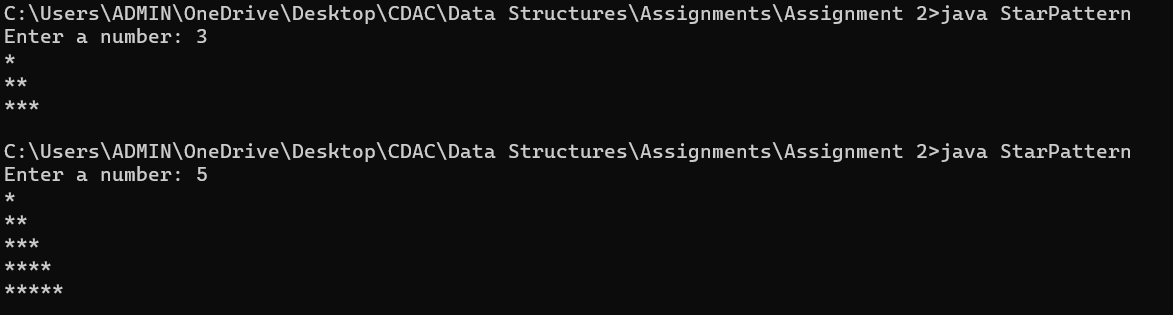
}

System.out.println();

}

}

}



2. **Remove Array Duplicates**

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

import java.util.\*;

public class RemoveArrayDuplicates{

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

System.out.print("Enter size of array: ");

int n = sc.nextInt();

int[] arr = new int[n];

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

{

System.out.print("Enter element "+ i+" : ");

arr[i] = sc.nextInt(); // Input array e.g. 1 1 2 2

}

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

// If array is already sorted, just add the last number from the group of repeated numbers to the new array i.e. only last occurence of that number

for(int i=0; i<n-1; i++)

{

if(arr[i] != arr[i+1]) // 1 != 1 false ; 1 != 2 true -> add last occurence of no. in new array; 2 != 2 false; won't go on last element of arr

arrayList.add(arr[i]);

}

arrayList.add(arr[n-1]); //adding last remaining element

System.out.println(arrayList.size());

/\* Another method using HashSet to store unique values

Integer[] arr = new Integer[n];

HashSet<Integer> set = new HashSet<>();

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

{

System.out.print("Enter element "+ i+" : ");

arr[i] = sc.nextInt();

set.add(arr[i]);

}

System.out.println(set.size());

\*/

}

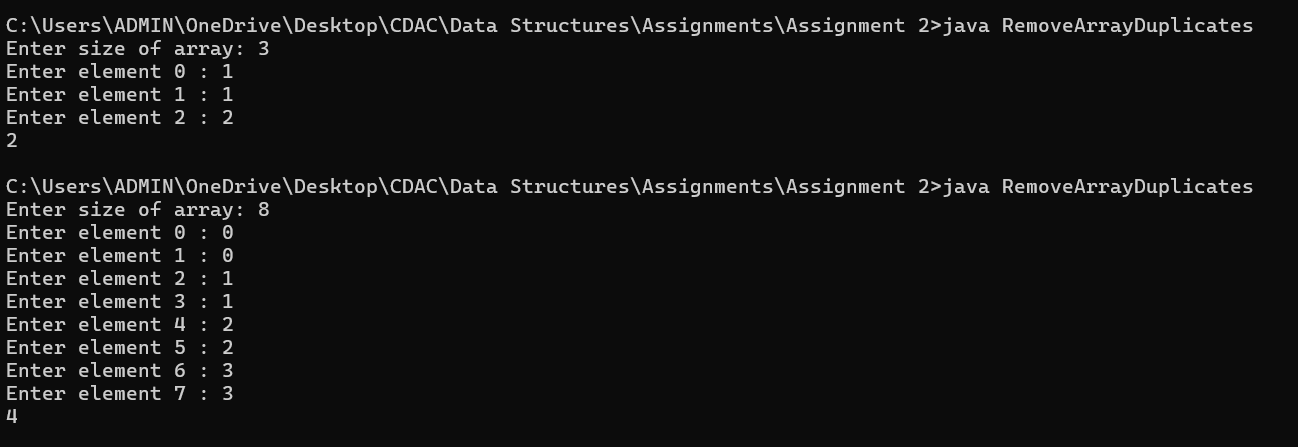
}

/\*

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.

import java.util.\*;

public class RemoveWhiteSpaces{

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

System.out.print("Enter string: ");

String str = sc.nextLine();

System.out.println(str.replace(" ","")); // replace() of String class replaces a character in the string with another character

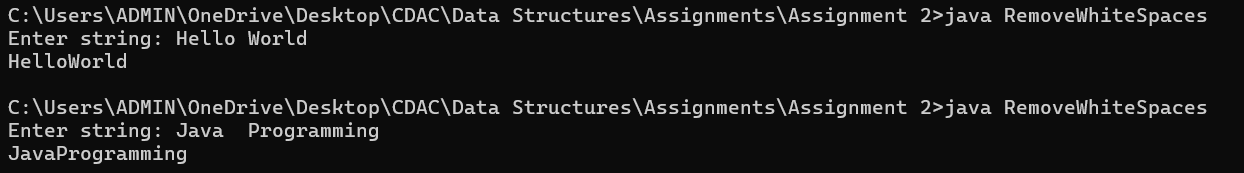
}

}

/\*Time Complexity: O(n)

Space Complexity: O(n)

\*/



4. **Reverse a String**

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

import java.util.\*;

public class ReverseString{

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

System.out.print("Enter string: ");

String str = sc.nextLine();

StringBuilder strB = new StringBuilder(str); // Created a StringBuilder string by passing it a String (we need reverse method of StringBuilder)

System.out.println(strB.reverse()); // Reversed the string using reverse() of StringBuilder

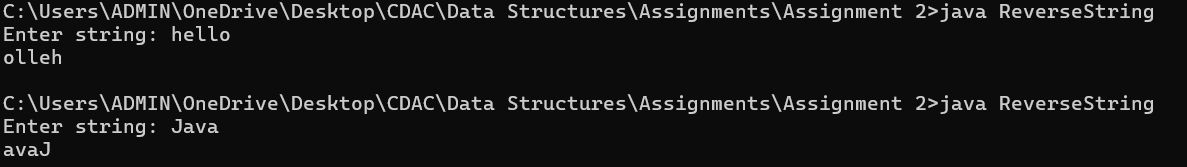
}

}

/\*Time Complexity: O(n)

Space Complexity: O(n)

\*/



5. **Reverse Array in Place**

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

import java.util.\*;

public class ReverseArray{

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

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

int n = sc.nextInt();

int[] arr = new int[n];

System.out.println("Enter elements of the array: ");

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

arr[i] = sc.nextInt();

}

for(int start = 0, end = n-1; start<end; start++, end--){ // Set start = 0 i.e. at starting index of array & end = n-1 i.e. ending index of the array

int temp = arr[start]; // Swapping logic to swap the first and last lement in the array

arr[start] = arr[end];

arr[end] = temp;

}

// Keep running the loop until start becomes greater than end

//E.g. 6 elements -> 0 < 5 1 < 4 2 < 3 3 < 2 X -> (even no. of elements) will stop when start becomes > end

//E.g. 5 elements -> 0 < 4 1 < 3 2 < 2 X -> (odd no. of elements) will stop when start is not less than end

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

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

}

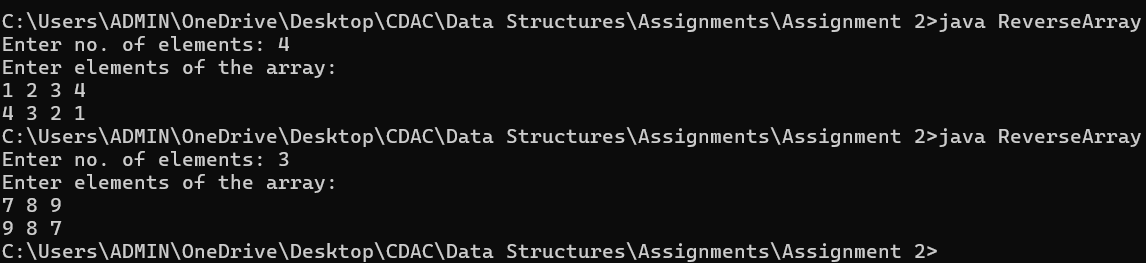
}

}

/\*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.

import java.util.\*;

public class ReverseWordsInString{

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

System.out.print("Enter string: ");

String str = sc.nextLine(); //Hello World

StringTokenizer stk = new StringTokenizer(str); //Tokenize the string (convert each word separated by space(default) into a separate string) using StringTokenizer

List<String> list = new ArrayList<>(); //ArrayList of strings to store the tokenized strings

while(stk.hasMoreTokens()){ //Check if there are tokens (words) available in the tokenized string

list.add(stk.nextToken()); // Add each token into the ArrayList of strings

}

for(int i= list.size() -1; i>=0 ; i--) // Print the ArrayList in reverse order starting from last index

System.out.print(list.get(i) + " ");

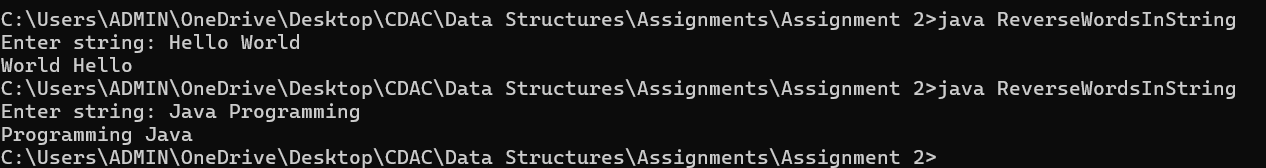
}

}

/\*Time Complexity: O(n)

Space Complexity: O(n)

\*/



7. **Reverse a Number**

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

import java.util.\*;

public class ReverseANumber{

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

System.out.print("Enter a number: ");

int n = sc.nextInt(); //1234

int lastDigit, reverseNum=0;

while(n!=0){

lastDigit = n%10; // Take last digit of number by % 10 (will get last digit as remainder) // 4; 3; 2; 1

reverseNum = (reverseNum \* 10) + lastDigit; // store that last digit in reverse number and then later keep multiplying by 10

// -> 0\*10 + 4 -> 4 ; 4\*10 +3 -> 43 ;

n/=10; // Keep dividing the original number by 10 -> 1234/10 -> 123 ...

}

System.out.println(reverseNum);

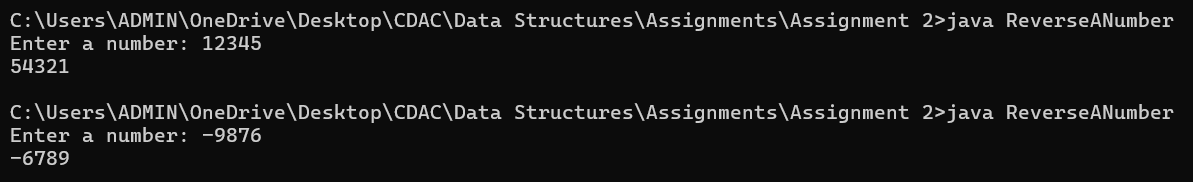
}

}

/\*Time Complexity: O(n) (n-> no. of digits in the number)

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.

import java.util.\*;

public class ArrayManipulation{

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

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

int n = sc.nextInt(); // 4

int[] arr = new int[n]; // 0 0 0 0

System.out.print("Enter no. of queries: ");

int q = sc.nextInt(); // 2

int[][] queries = new int[q][3]; // [[1, 3, 50], [2, 4, 70]]

// Getting the queries O(q\*3)-> O(q)

System.out.print("Enter queries elements: ");

for(int row=0; row<queries.length; row++){

for(int col=0; col<3; col++){

queries[row][col] = sc.nextInt();

}

}

// Inputing the values from queries into array O(q\*n)

for(int i=0; i<q; i++){ // 0 to 1

for(int j = queries[i][0] - 1; j < queries[i][1]; j++){ // 0 1 2 ;

arr[j] = arr[j] + queries[i][2];

}

}

/\*

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

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

}

\*/

Arrays.sort(arr); //Sort array O(n log n)

System.out.println(arr[arr.length-1]); // Print last value (i.e. the max)

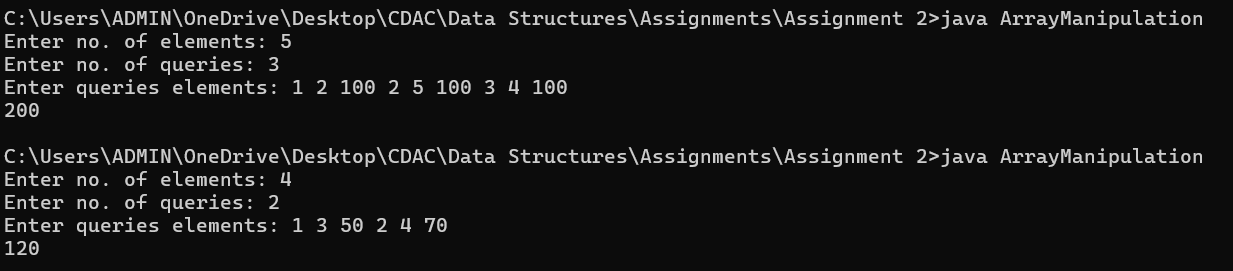
/\*Time Complexity: O(q\*n) -> q for the loop that runs to take queries and n for loop to calculate max value

Space Complexity: O(n)

\*/

}

}



**Optimized Version**

**(Using prefix sum array technique i.e. we don’t need to iterate to add value at all indexes. Instead, just add the value at starting index and -value at ending index + 1 position)**

import java.util.\*;

public class ArrayManipulationOptimized{

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

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

int n = sc.nextInt(); // 4

int[] arr = new int[n]; // 0 0 0 0

System.out.print("Enter no. of queries: ");

int q = sc.nextInt(); // 2

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

int startIndex = sc.nextInt() - 1 ; //1-1 = 0 <- since we start from 0th index

int endIndex = sc.nextInt() - 1; //3-1 = 2

int value = sc.nextInt(); // 3

// We are using method of adding the value just at the start index and assuming that next 0s are that value as well

// We will set -value to indicate that the value is note present from here on out and it needs to be removed from sum that we will be calculating

arr[startIndex] = arr[startIndex] + value; //add value at startindex e.g. 3 0 0

if(endIndex < n-1){ // because if we are going to add -value at endindex+1, endindex should be less than the last index else it will go out of bond

arr[endIndex+1] = arr[endIndex+1] - value; //add -value at endindex+1 e.g. 3 0 0 -3 to indicate that value was not added from here on

// Since we are going to add the values and store in a variable currentSum, there needs to be a -value so that value gets removed from the currentSum

}

}

int max = Integer.MIN\_VALUE;

int currentSum = 0;

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

currentSum = currentSum + arr[i];

if(currentSum > max){

max = currentSum;

}

}

System.out.println(max);

}

}

/\*Time Complexity: O(q+n) -> q for the loop that runs to take queries and n for loop to calculate max value

Space Complexity: O(n)

\*/

9. **String Palindrome**

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

import java.util.\*;

public class StringPalindrome{

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

System.out.print("Enter a string: ");

String str = sc.nextLine();

boolean isPalindrome = true;

for(int start=0, end=str.length()-1; start < end; start++, end--) //O(n/2) -> O(n)

{

if(str.charAt(start) != str.charAt(end)) // Comparing characters from both ends of the string

{

isPalindrome = false; // set flag to false if starting and ending characters are not equal

break;

}

}

if(isPalindrome)

{

System.out.println("true");

}

else

System.out.println(false);

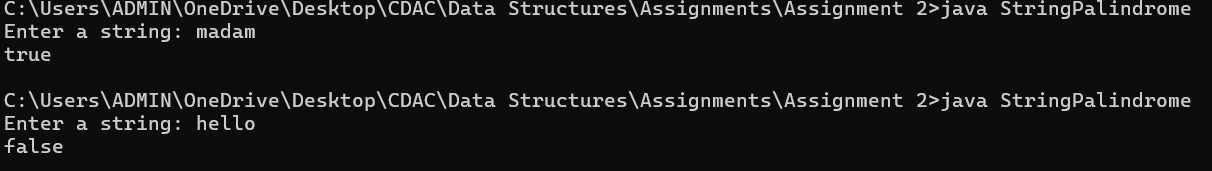
}

}

/\*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.

import java.util.\*;

public class ArrayLeftRotation{

static void reverseArray(int[] arr, int i, int n){

int temp;

for(int start=i, end=n-1; start<end; start++, end--){

temp = arr[start];

arr[start] = arr[end];

arr[end] = temp;

}

}

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

System.out.print("Enter no. of elements ");

int n = sc.nextInt();

int[] arr = new int[n];

System.out.print("Enter elements ");

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

arr[i] = sc.nextInt();

}

System.out.print("By how many positions do you want to shift the array to the left? ");

int d = sc.nextInt();

d = d % arr.length; // Because if we shift the elements by d (e.g. 6) then shifting by 6%5=>1 would give the same array

// 1 2 3 4 5

// Shifting left by 6->2 3 4 5 1 and Shifting left by 1 -> 2 3 4 5 1

// Also, this helps the code work if array size is less than the positions to be shifted

reverseArray(arr, 0, n); // Reverse the entire array first -> 5 4 3 2 1

// if d = 2 -> 2 % 5 -> 2 0 2

reverseArray(arr, 0, n-d); // Reverse the array from 0th index to n-d-1 index -> 3 4 5 2 1

// 3 4

reverseArray(arr, n-d, n); // Reverse the array from n-d index to n-1 index -> 3 4 5 1 2

// Hence, we got the array shifted left by 2 places

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

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

}

}

}

/\*

Time Complexity: O(n)

Space Complexity: O(n)

\*/

