

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:

*

**

PROGRAM:

//Printing Pattern

import java.util.Scanner;

```
class Pattern{
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number: ");
        int n = sc.nextInt();

        for(int i = 1; i<=n; i++){
            for(int j = 0; j<i; j++){
                System.out.print("*");
            }
            System.out.println(" ");
        }
    }
}
```

OUTPUT:

```
C:\Dac\ADS\Practice>java Pattern
Enter number: 3
*
**
***

C:\Dac\ADS\Practice>java Pattern
Enter number: 5
*
**
***
****
*****
```

EXPLANATION:

Here first the input(n) will taken from user. This n will determines the number of rows in the triangle pattern.

The outer loop runs from $i=1$ to $i = n$, for each iteration of this loop it will print new line after printing I no. of *

The inner loop runs from $j=0$ to $j < i$, printing i number of * in each row. This means that row 1 has 1 * , row 2 has 2*.

TIME COMPLEXITY: $O(n^2)$

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.

Test Cases:

Input: arr = [1, 1, 2]

Output: 2

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

Output: 4

PROGRAM:

```
//Remove Array Duplicate
import java.util.Scanner;

class RemoveArrayDuplicate{
```

```

static int removeDuplicate(int[] arr)
{
    if(arr.length == 0)
        return 0;

    int uniqueIndex = 0;

    for(int i = 1; i < arr.length; i++)
    {
        if(arr[i] != arr[uniqueIndex])
        {
            uniqueIndex++;
            arr[uniqueIndex] = arr[i];
        }
    }
    return uniqueIndex + 1;
}

public static void main(String args[]){
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter Array size: ");
    int n = sc.nextInt();

    int[] arr = new int[n];

    System.out.print("arr = ");
    for(int i = 0; i < n; i++){
        arr[i] = sc.nextInt();
    }

    int newLength = removeDuplicate(arr);
    System.out.println(newLength);
    sc.close();
}
}

```

OUTPUT:

```

C:\Dac\ADS\Practice>java RemoveArrayDuplicate
Enter Array size: 3
arr =  1 1 2
2

C:\Dac\ADS\Practice>java RemoveArrayDuplicate
Enter Array size: 8
arr =  0 0 1 1 2 2 3 3
4

```

EXPLANATION:

At start program takes an array size n and then reads the array elements.
Then the method removeDuplicate() modifies the input array in place by moving unique elements to the front.

After that unique elements are compared to each other in sequence, and the first occurrence of each unique element is retained.

Then the output is printed.

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.

Test Cases:

Input: "Hello World"

Output: "HelloWorld"

Input: " Java Programming "

Output: "JavaProgramming"

PROGRAM:

```
import java.util.Scanner;

class RemoveWhiteSpace{
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter String : ");

        String str = sc.nextLine();

        String newStr = str.replaceAll("\\s+", "");

        System.out.println(newStr);

        sc.close();
    }
}
```

OUTPUT:

```
C:\Dac\ADS\Assignments\Assignment2>java RemoveWhiteSpace
Enter String : Hello World
HelloWorld

C:\Dac\ADS\Assignments\Assignment2>java RemoveWhiteSpace
Enter String : Java Programming
JavaProgramming
```

EXPLANATION:

At first the program will take input string from user.

Then the `replaceAll()` method is called on the input string `str`, where, the regular expression `\\s+` matches all occurrences of one or more whitespace characters (including spaces, tabs, newlines, etc.).

These whitespace characters are replaced with an empty string `""`, effectively removing them from the string, then modified string `newStr`, which no longer contains white spaces, is printed.

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"

PROGRAM:

```
import java.util.Scanner;

class ReverseString{
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter String : ");

        String str = sc.nextLine();

        char[] chArr = str.toCharArray();

        for(int i = chArr.length-1; i >= 0; i--){
            System.out.print(chArr[i]);
        }

        sc.close();
    }
}
```

```
}  
}
```

OUTPUT:

```
C:\Dac\ADS\Assignments\Assignment2>java ReverseString  
Enter String : Hello  
olleH  
C:\Dac\ADS\Assignments\Assignment2>java ReverseString  
Enter String : Java  
avaJ
```

EXPLANATION:

At first the program will take input string from user.

Then string str is converted to a character array chArr using the toCharArray() method, which allows to access individual characters in the string, then loop starts from chArr.length - 1 & continues until i >= 0. During each iteration, the character at the current index i is printed & inside the loop, each character is printed in reverse order.

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]

PROGRAM:

```
import java.util.Scanner;  
  
class ReverseArray{  
    public static void main(String args[]){  
        Scanner sc = new Scanner(System.in);  
        System.out.print("Enter Size : ");  
        int n = sc.nextInt();  
  
        int[] arr = new int[n];  
        System.out.print("Enter Element : ");  
        for(int i = 0; i < arr.length; i++){  
            arr[i] = sc.nextInt();  
        }  
    }  
}
```

```

        System.out.print("[");
        for(int j = arr.length-1; j>=0; j--){
            System.out.print(arr[j]);
            if (j > 0) {
                System.out.print(", ");
            }
        }
        System.out.println("]");
        sc.close();
    }
}

```

OUTPUT:

```

C:\Dac\ADS\Assignments\Assignment2>java ReverseArray
Enter Size : 4
Enter Element : 1 2 3 4
[4, 3, 2, 1]

C:\Dac\ADS\Assignments\Assignment2>java ReverseArray
Enter Size : 3
Enter Element : 7 8 9
[9, 8, 7]

```

EXPLANATION:

At first input will be taken by the user then as here we need to reverse the array so I use for loop to iterate through the array in reverse order, starting from the last element (arr.length - 1) and moving towards the first element (0).

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"

PROGRAM:

```
import java.util.Scanner;
```

```
class ReverseWord{
```

```

    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Sentence : ");
        String str = sc.nextLine();
        String[] newStr = str.split(" ");

        for(int i = newStr.length-1; i>=0; i--){
            System.out.print(newStr[i]+ " ");
        }

        sc.close();
    }
}

```

OUTPUT:

```

C:\Dac\ADS\Assignments\Assignment2>java ReverseWord
Enter Sentence : Hello World
World Hello
C:\Dac\ADS\Assignments\Assignment2>java ReverseWord
Enter Sentence : Java Programming
Programming Java
C:\Dac\ADS\Assignments\Assignment2>

```

EXPLANATION:

At first the input will be taken from user after that I have use the `str.split()` method that returns an array `newStr` where each element is a word from the sentence. Then a loop iterates over the array `newStr` in reverse order, starting from the last word (`newStr.length - 1`) and ending with the first word (0). During each iteration, the word at the current index is printed, followed by a space & the program ends after printing the sentence with reversed word order.

TIME COMPLEXITY: $O(n)$

SPACE COMPLEXITY: $O(n)$

7. Reverse a Number

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

Test Cases:

```

Input: 12345
Output: 54321
Input: -9876
Output: -6789

```

PROGRAM:

```

import java.util.Scanner;

```



```

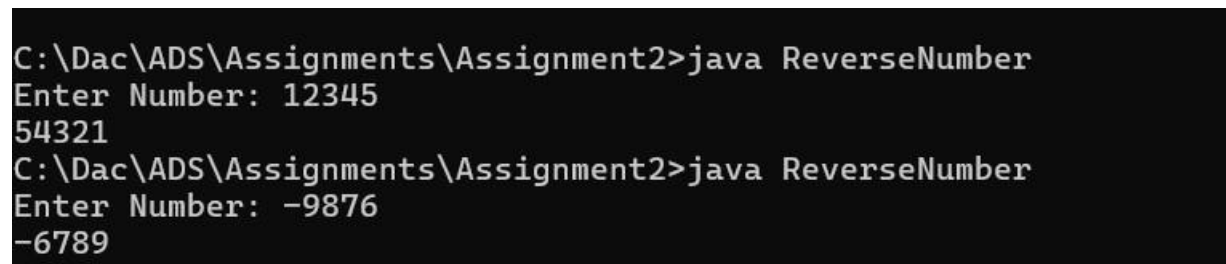
public class ReverseNumber{
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Number: ");
        int n = sc.nextInt();

        int result = 0;

        while( n != 0 ){
            int rem = n % 10;
            n = n/10;
            result = result * 10 + rem;
        }
        System.out.print(result);
    }
}

```

OUTPUT:



```

C:\Dac\ADS\Assignments\Assignment2>java ReverseNumber
Enter Number: 12345
54321
C:\Dac\ADS\Assignments\Assignment2>java ReverseNumber
Enter Number: -9876
-6789

```

EXPLANATION:

The input is taken by the user then a variable result is initialized to 0, which will hold the reversed number.

The loop runs until (n !=0). For each iteration:

The remainder rem is calculated using $n \% 10$ to get the last digit of n.

The number n is updated by dividing it by 10 ($n = n / 10$) to remove the last digit.

The reversed number result is updated using $result = result * 10 + rem$.

Once n becomes 0, the loop ends & the result is printed.

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.

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

PROGRAM:

```
public class ArrayManipulation {
    public static long arrayManipulation(int n, int[][] queries) {
        long[] array = new long[n + 1]; // create an array of n+1 elements

        // Apply the difference array concept
        for (int[] query : queries) {
            int a = query[0] - 1; // converting to 0-based indexing
            int b = query[1];     // range is inclusive, so no need to adjust
            int k = query[2];

            array[a] += k;
            if (b < n) {
                array[b] -= k;
            }
        }

        // Calculate the maximum value after applying all range updates
        long max = 0;
        long current = 0;
        for (int i = 0; i < n; i++) {
            current += array[i];
            if (current > max) {
                max = current;
            }
        }

        return max;
    }

    public static void main(String[] args) {

        int n1 = 5;
        int[][] queries1 = {{1, 2, 100}, {2, 5, 100}, {3, 4, 100}};
        System.out.println(arrayManipulation(n1, queries1));

        int n2 = 4;
        int[][] queries2 = {{1, 3, 50}, {2, 4, 70}};
        System.out.println(arrayManipulation(n2, queries2));
    }
}
```

OUTPUT:

C:\Dac\ADS\Assignments\Assignment2>java ArrayManipulation

200

120

EXPLANATION:

Firstly created an array of size $n + 1$ to account for the difference array technique.

For each query $[a, b, k]$, add k at the start index a (adjusted for 0-based index) and subtract k at $b+1$ (to remove the effect after the end of the range).

Then calculate the prefix sum to get the final values in the array. The maximum value encountered during this process is the result.

TIME COMPLEXITY: $O(n + m)$ where n is the size of the array and m is the number of queries.

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

Here's a continuation of the list of assignment questions starting from question 21, with two test cases for each:

PROGRAM:

```
import java.util.Scanner;
```

```
class StringPalindrome{
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter String : ");

        String str = sc.nextLine();
        str = str.toLowerCase();

        char[] chArr = str.toCharArray();
        boolean isPalindrome = true;

        for(int i = chArr.length-1; i >= 0; i--){
            if (str.charAt(i) != str.charAt(chArr.length - 1 - i)) {
                isPalindrome = false;
                break;
            }
        }
        if(isPalindrome)
            System.out.print("true");
        else
            System.out.print("false");
        sc.close();
    }
}
```

OUTPUT:

```
C:\Dac\ADS\Assignments\Assignment2>java StringPalindrome
Enter String : Madam
true
C:\Dac\ADS\Assignments\Assignment2>java StringPalindrome
Enter String : Hello
false
```

EXPLANATION:

Firstly the input will be taken from the user then the input is converted into lower case after that loop will execute until the condition gets false then the output will print.

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.

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]

FLOWCHART:**PROGRAM:**

```
import java.util.Scanner;

class RotateArray{
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Size: ");
        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("Enter d: ");
        int d = sc.nextInt();
```

```

        for(int i = 0; i<d; i++){
            int first = arr[0];
            for(int j = 0; j<arr.length-1; j++){
                arr[j] = arr[j+1];
            }
            arr[arr.length-1] = first;
        }
        //System.out.print();
        for(int i = 0; i<arr.length;i++){
            System.out.print(arr[i]+" ");
        }
    }
}

```

OUTPUT:

```

C:\Dac\ADS\Assignments\Assignment2>java RotateArray
Enter Size: 5
Enter Elements: 1 2 3 4 5
Enter d: 2
3 4 5 1 2
C:\Dac\ADS\Assignments\Assignment2>java RotateArray
Enter Size: 4
Enter Elements: 10 20 30 40
Enter d: 1
20 30 40 10

```

EXPLANATION:

Firstly the input is taken from user for size of array elements of array & how many time the array should rotate then the loop will execute until the condition returns false & output is printed.

TIME COMPLEXITY: $O(n)$

SPACE COMPLEXITY: $O(1)$