classmate
Date 26/09/24
Page

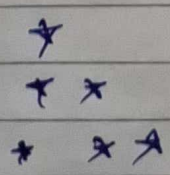
Assignment-2

Q.1)
Printing Patterns

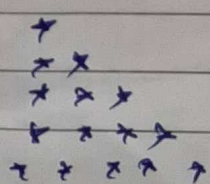
1) WAJP to print patterns such as right angle of stars

```
class Recpattern {
    static void patt(int n){
        if(n==0)
        {
            return n;
        }
        for(int i=0 ;i<n ;i++)
        {
            for(int j=0 ; j<=i ;j++)
            {
                System.out.println("*");
            }
            System.out.println("");
        }
    } public static void main(String[] args)
    {
        patt(3);
    }
}
```

Output=



if n=5



Explaination -

Main method calls patt(3), passing n=3 as argument to method patt(int n).

pt it check ib(n==0), not matchn doesn't ~~print~~ return. & execution Continueu

2nd step
first for loop stat with i=1 & i<n & 2nd run after checking condn print *.

After inner loop finishes, moves cursor to nxt line.

Time Complexity - $O(n^2)$

Space Complexity is - O(1)
[variable i, j, n all are integer variable, takeup const. space.

NPCI

Q2) Remove Array Duplicates

WAJP to remove duplicates from a sorted array & 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]

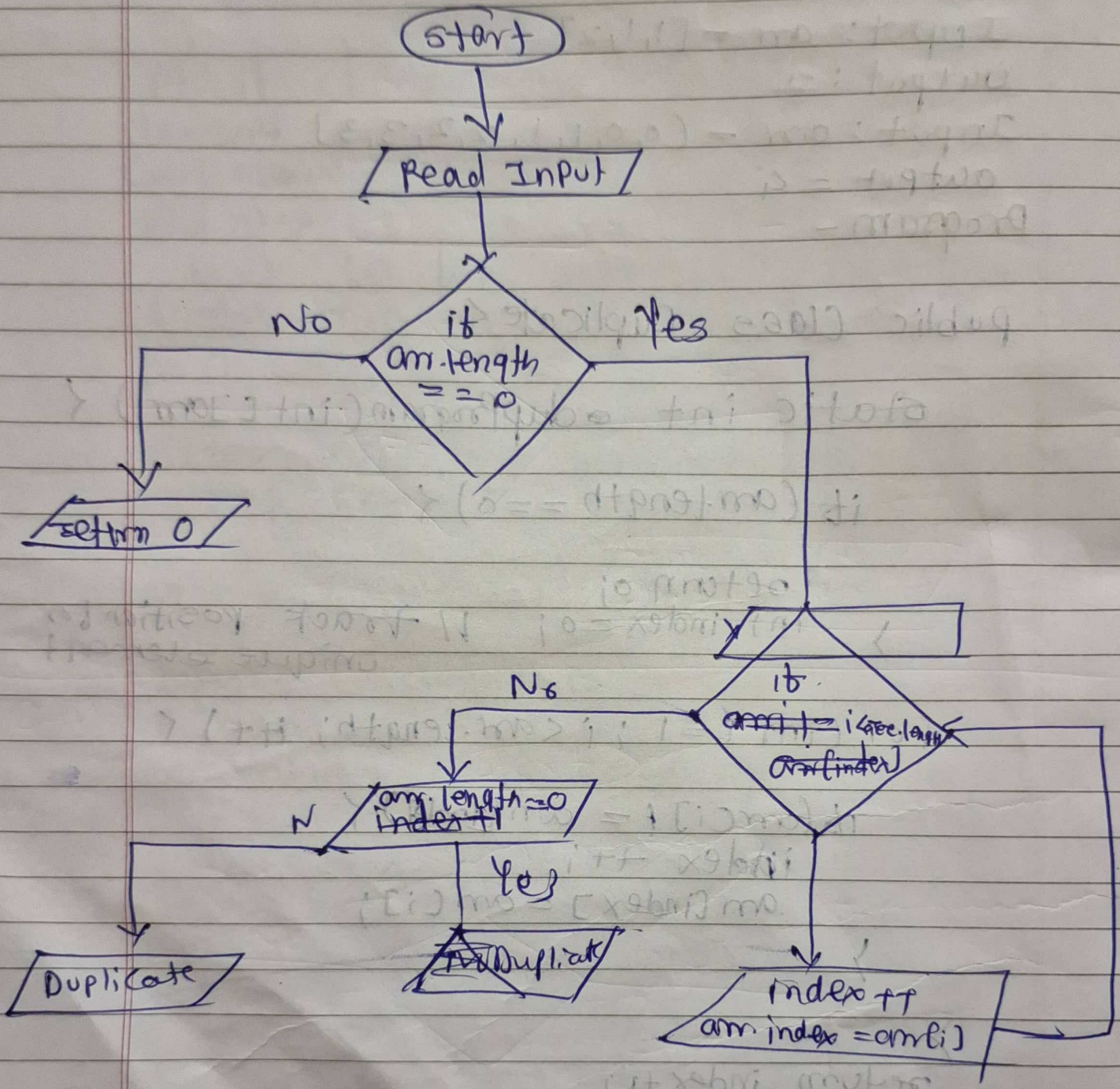
output = 4

Program -

```
public class Duplicate {
    static int dupProgram(int[] arr) {
        if (arr.length == 0) {
            return 0;
        }
        int index = 0;  // track position for unique element
        for (int i = 1; i < arr.length; i++) {
            if (arr[i] != arr[index]) {
                index++;
                arr[index] = arr[i];
            }
        }
        return index + 1;
    }
    psvm(String args[]) {
        int[] arr1 = {1,1,2}
        int[] arr2 = {0,0,1,1,2,2,3,3};
        s.o.pln("Input" + Arrays.toString(arr1));
        s.o.pln("output:" + removeDuplicates(arr1));
    }
}
```

Output = 2

Explaination —



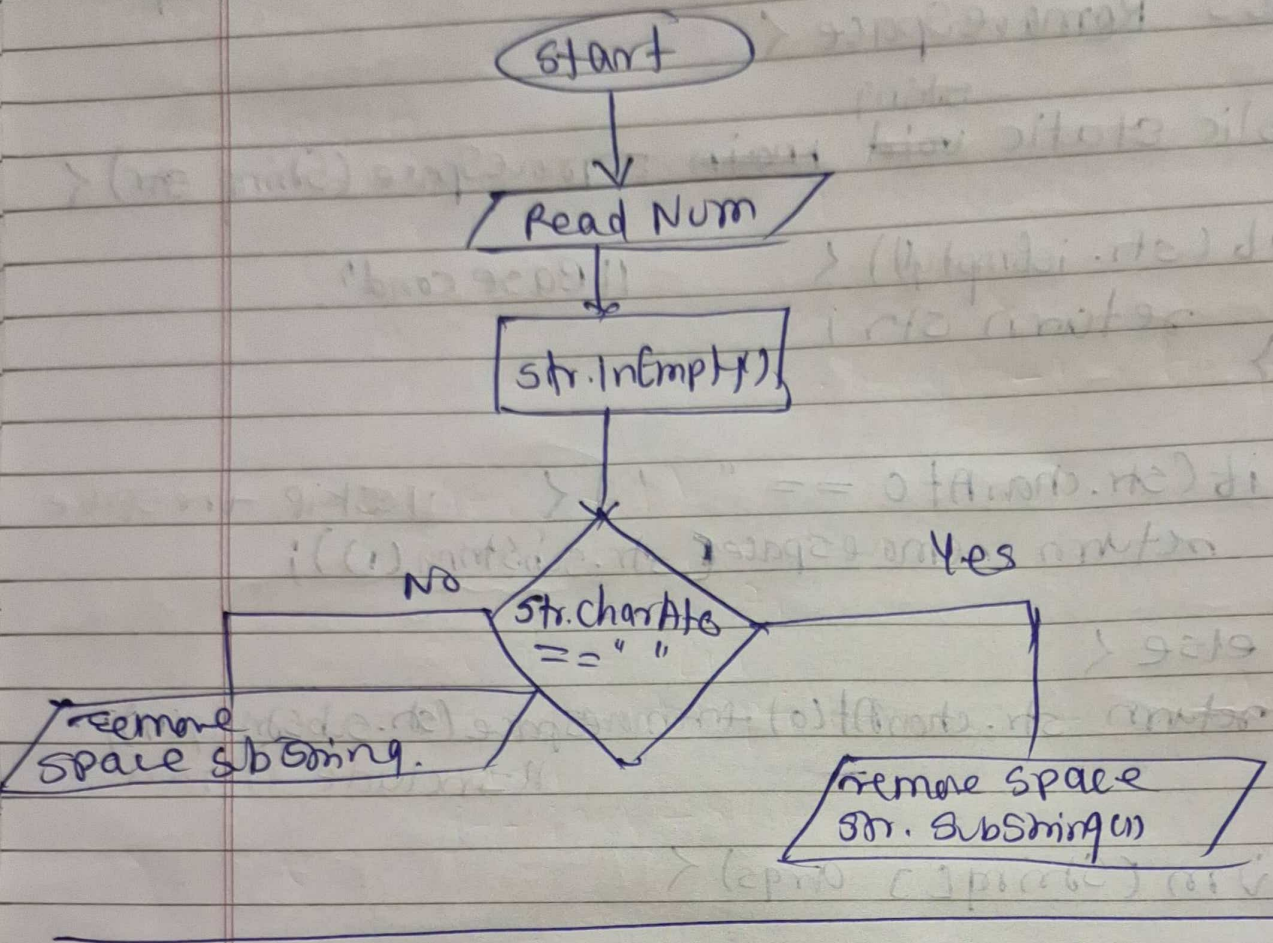
Q.3) WAJP to Remove White spaces from string

Program -

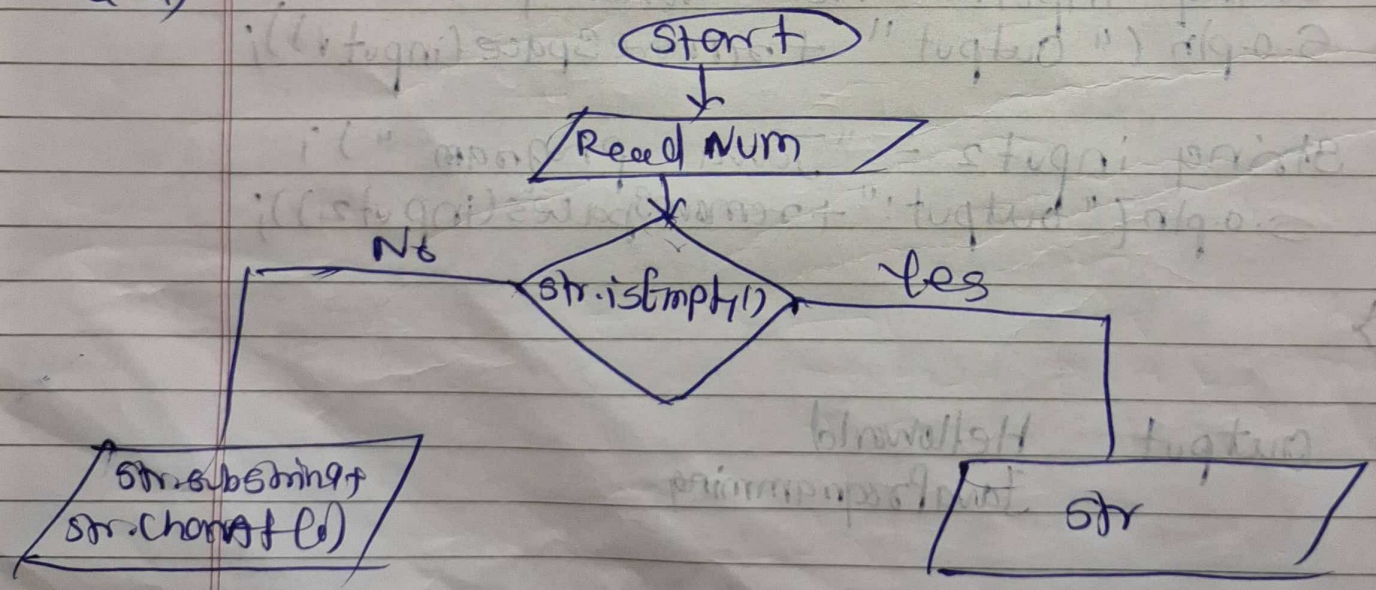
```
class RemoveSpace {
public static void main removeSpace(String str) {
    if (str.isEmpty()) {          // Base cond^n
        return str;
    }
    if (str.charAt(0) == ' ') {   // skip the space
        return removespace(str.substring(1));
    }
    else {
        return str.charAt(0) + removespace(str.substring(1));
    }                             // Include character.
}
psvm (String[] args) {
    String input1 = "Hello World";
    S.o.pln ("Output" + removeSpace(input1));
    String input2 = "Java program";
    S.o.pln("output:" + removeSpaces(input2));
}
}
```

Output HelloWorld
JavaProgramming

Q.3)


Start
Read Num
str.isEmpty()
str.charAt(0) == " "
No
Yes
remove space substring.
remove space str.substring(1)

Q.4)


Start
Read Num
str.isEmpty()
No
Yes
str.substring(1) str.charAt(0)
str

Q-4) Reverse a string

program - write a java program to reverse a given string.

Test Cases -

Input "hello"
output "olleh"

Program -

```
class Reverse~~String~~{

public static String reverse(String str){

if(str.isEmpty()){
    return str;
}
    return reverse(str.substring(1)+ str.charAt(0);
}
psvm(String[] args){

    String input1 = "hello";
    S.o.pln("output" + reverse(input1));

}
}
```

output olleh.

Q.5]

```
class A
{
  public static void main (String[] args){

    int a[] = {1,2,3,4,5,6};
    int i = 0 ; j = a.length-1 , temp;

    while(i<j)
    {
       temp = a[i];
      a[i] = a[j]
      a[j] = temp;
       i++;
       j--;
  }
S.O.pln (Arrays.toString(a));
 }
}
```

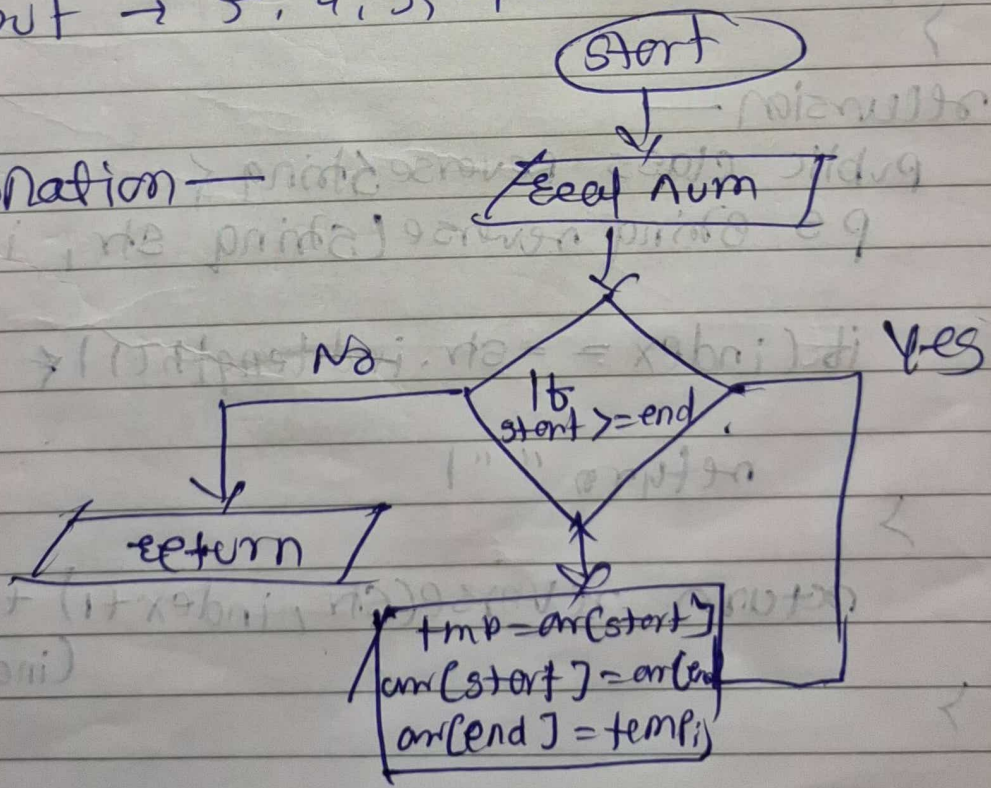
Using recursion →

```
Class Reverse{
public static void reverse(int[] arr,
                           int start, int end){

    if (start >= end){
        return;
    }
```

```
    int tmp = arr[start];
    arr[start] = arr[end];
    arr[end] = temp;

    reversearray(arr, start+1, end-1);
}
perm(String[] args){

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

S.o.pln("original ~~Array~~" + ~~Arrays.toString~~(arr));

  reverseArray(arr, 0, arr.length-1);
  S.o.pln("Reversed Array :" + Arrays.toString
                              (arr));
}
}
```

output → 5, 4, 3, 2, 1

Explaination —



6) Reverse words in a String.

Simple

```
class ReverseAString {

    psvm(String args[])
    {
        String name = "Hello world";

        int leng = name.length();
        String rev = "";

        for(int i = leng-1;i>=0;i--)
        {
            rev = rev+name.charAt(i);
        }
        S.o.pln("Reverse of "+name+" is "+rev);
    }
}
```

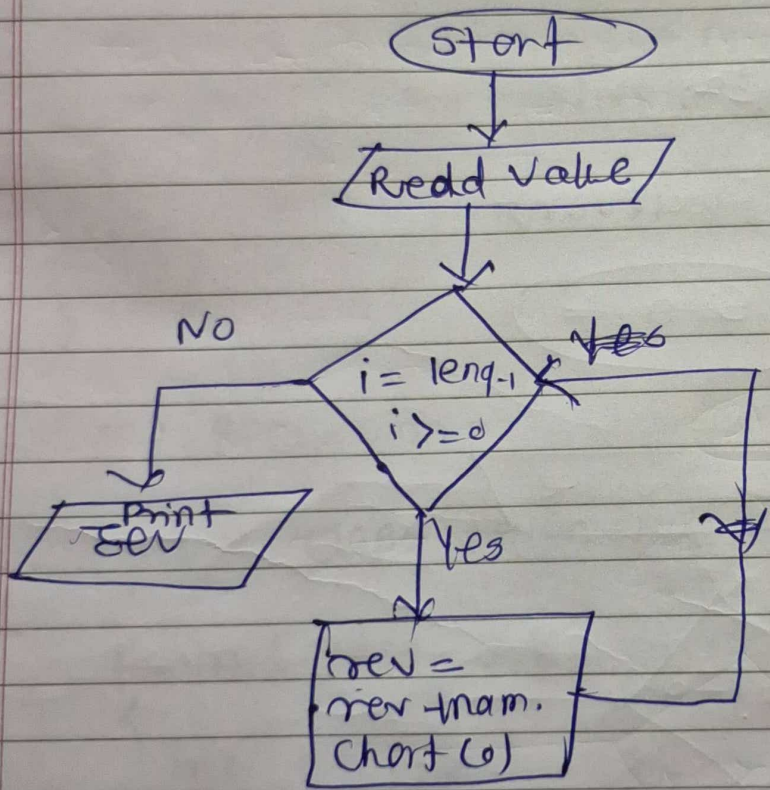
Using recursion —

```
public class ReverseString {
    ps String reverse(String str, int index){

    if(index == str.length()){
        return "";
    }
    return reverse(str, index+1) + str.charAt
                                    (index);
}
```

```
public static void main(String [] args){
String input = "Hello";
S.o.pln (Reverse(input, 0));
```

Output olleh

Flow Chart



7) Reverse Number —

```
Class ReverseNumber {

    psvm(String args[])
    {
        int no = 5434, rem, rev = 0;

        while (no != 0)
        {
            rem = no % 10;
            rev = rev * 10 + rem;
            no = no / 10;
        }
        S.o.pln(rev);
    }
}
```

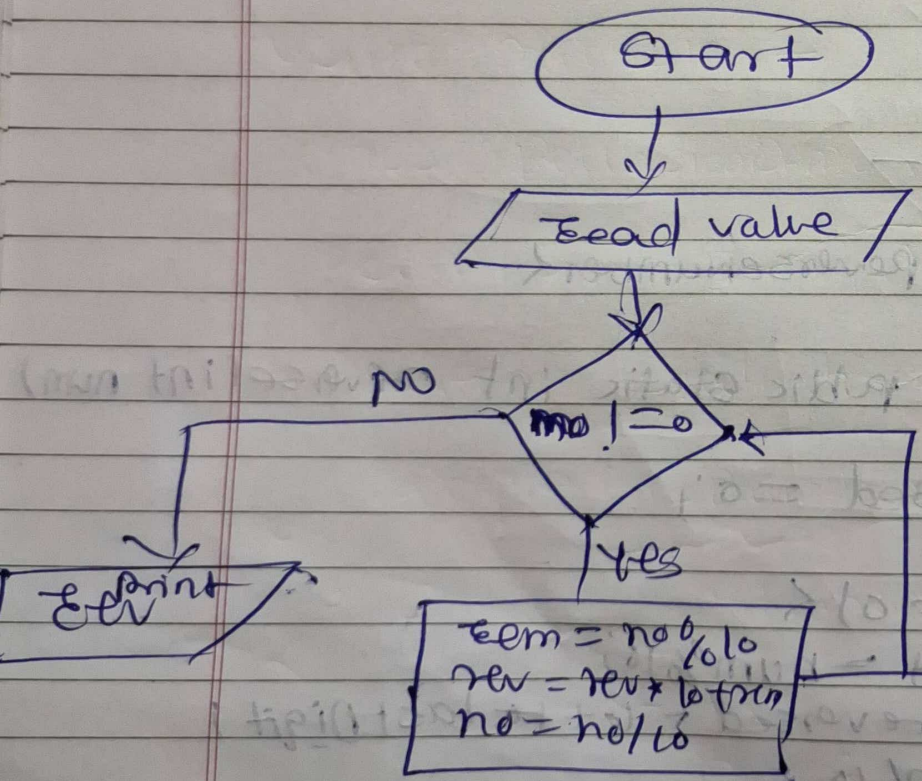
Using Recursion —

```
public class ReverseNumber{

psvm(String public static int reverse(int num)
{
    int reversed = 0;

while(num != 0){
int lastdigit = num % 10;
reversed = reversed * 10 + lastDigit;
num = num / 10;
}
```

```
return reversed;
}

psvm(String arg[]){

  int input = 12345;

   S.o.pln("Input" +input);

  S.o.pln(reverse(input));
}
}
```

Output — 54321

Flowchart



Q.9] ~~class~~

```
class RevString {

public static void main (String[] args)
{
    String s = "Hello";
    String rev = "";

    for(int i = s.length()-1; i>=0; i--)
    {
        rev = rev+s.charAt(i);
    }
    S.o.pln(rev);
}
}
```

using recursion -

```
class RecursionDemo {

 psvm(String[] args) {

   boolean isPalindrome = isPalindrome(0,4,"level");

    S.o.pln(isPalindrome);
}
p s boolean isPalindrome(int i, int j, String str)
{
     if(i>=j){
      return true;
}
```

```
if (str.charAt(i) != str.charAt(j)) {
        return false;
    }
    i = i+1;
    j = j-1;

    return isPalindrome(i, j, str);
}
}
```

Output false

Explaination –