C-DAC Mumbai

Date 25/09/2024

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. Armstrong Number
Problem: Write a Java program to check if a given number is an Armstrong number.

Test Cases:

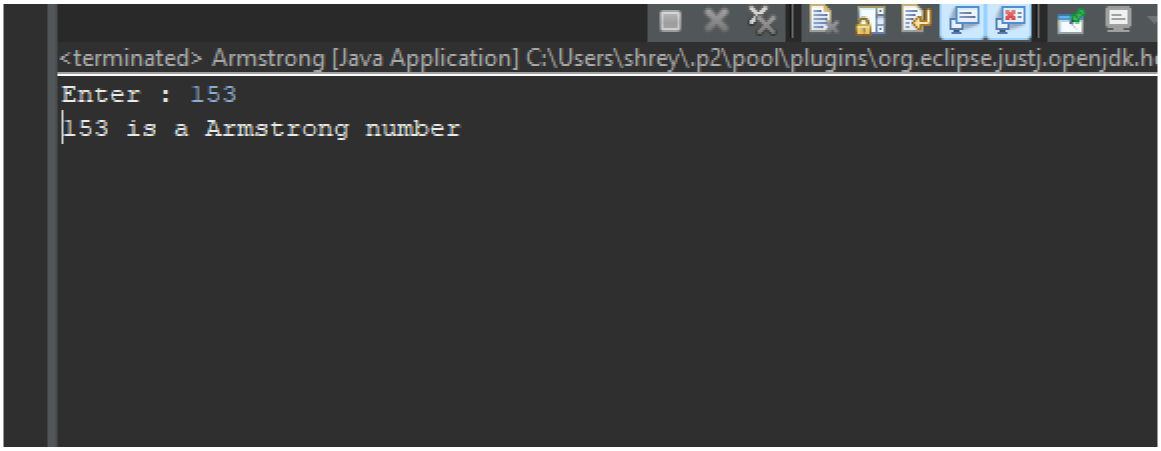
Input: 153
Output: true
Input: 123
Output: false

Sol:

```
import java.util.Scanner;
public class Armstrong {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter : ");
        int n = sc.nextInt();
        int m=n, count=0, t=0, ans=0
        while (m > 0) {
            m = m / 10;
            count++;
        }
        m = n;
        for (int i = 0; i < count; i++) {
            t = m % 10;
            ans += (int) Math.pow(t, count);
            m =  m / 10;
        }

        if ( ans == n ) {
            System.out.println(n + " is a Armstrong number");
        }
        else {
            System.out.println(n + " is not a Armstrong number");
```

```
                    }
          }
}
```

O/P:



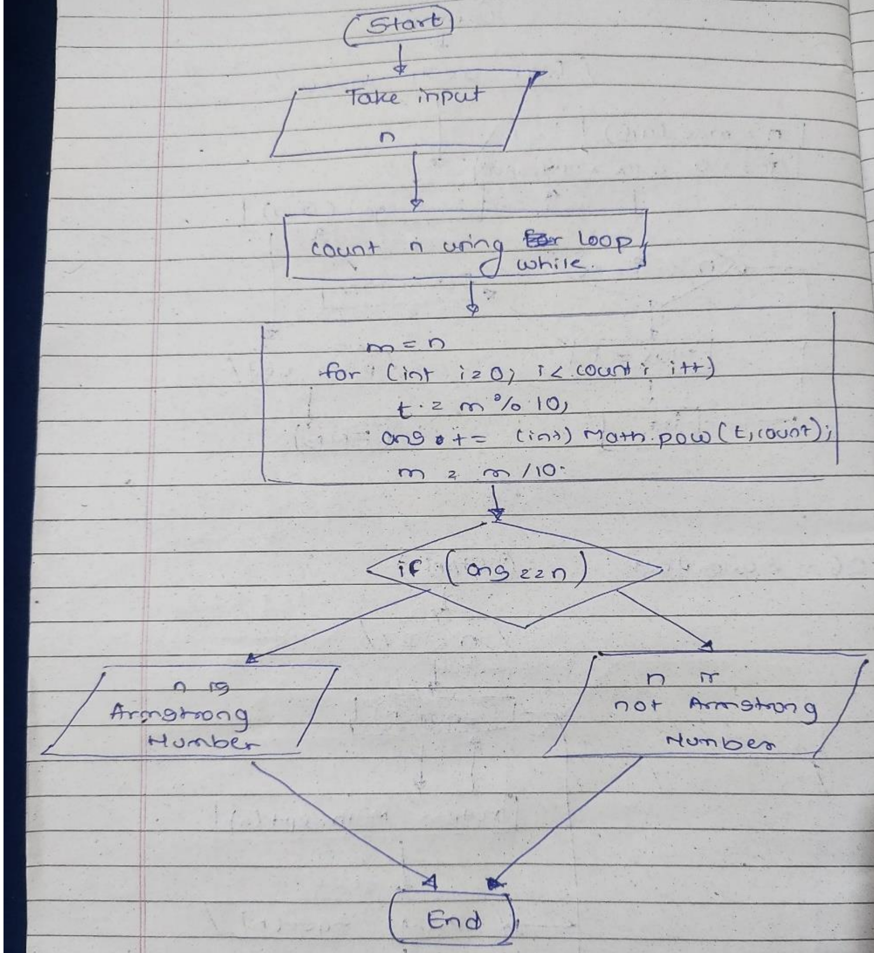
<terminated> Armstrong [Java Application] C:\Users\shrey\.p2\pool\plugins\org.eclipse.justj.openjdk.h

```
Enter : 153
153 is a Armstrong number
```

FlowChart:

01 Armstrong Num

02 -



```
m = n
for (int i=0; i< count; i++)
    t = m % 10;
    ans += (int) Math.pow(t,count);
    m = m /10.
```

2. Prime Number
Problem: Write a Java program to check if a given number is prime.

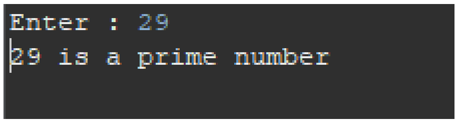
Test Cases:

Input: 29
Output: true
Input: 15
Output: false

SOL:

```
public class Prime_No {
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter : ");
		int n = sc.nextInt(),ans=1;
		if ( n <= 1) {
			ans = 0;
		}
		else if ( n == 2 || n == 3 ) {
			ans = 1;
		}
		for (int i = 2; i <= Math.sqrt(n); i++) {
			if ( n % i == 0) {
				ans = 0;
				break;
			}
		}

		if (ans == 1) {
			System.out.println(n+" is a prime number");
		}
		else {
			System.out.println(n+" is not a prime number");
		}
	}

}
```

O/P:

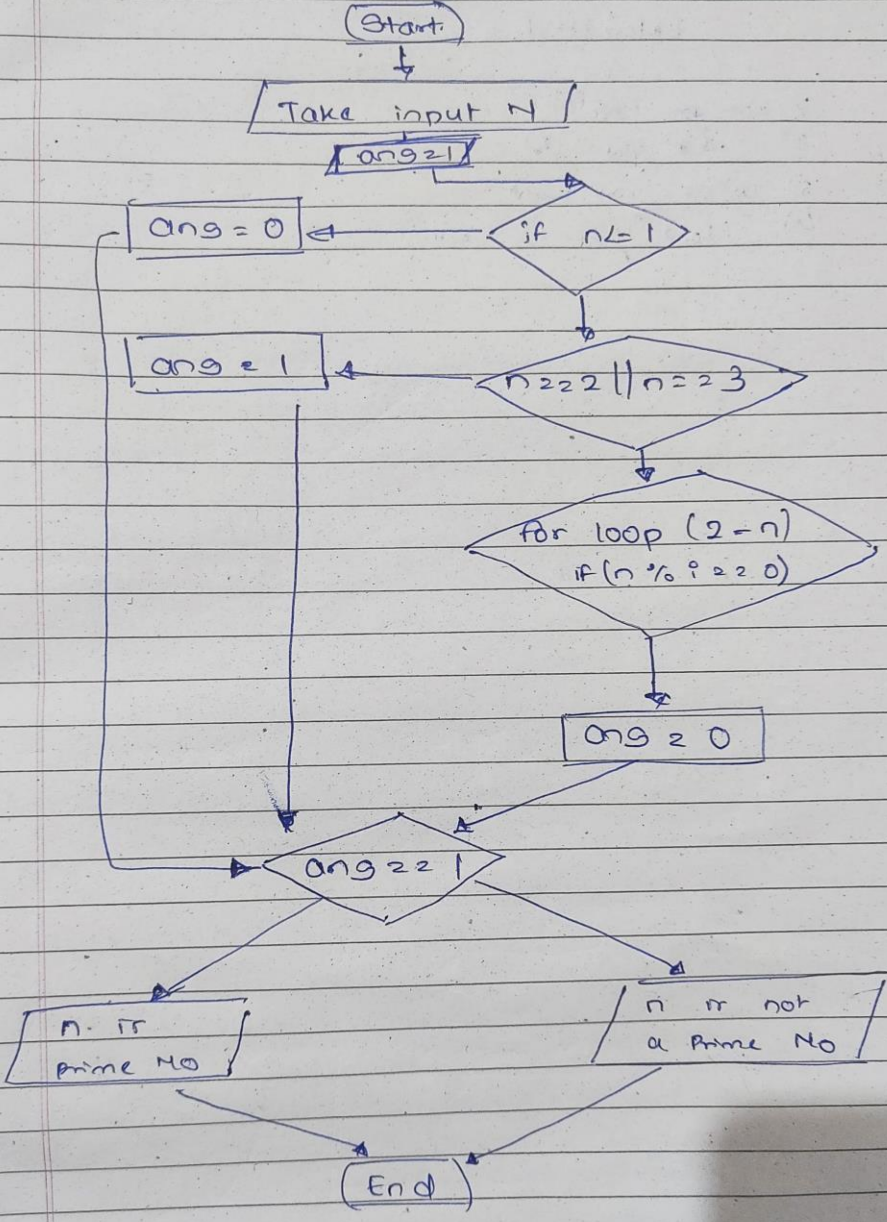


```
Enter : 29
29 is a prime number
```

FlowChart :

Page No. ____ Date ____

02 - Prime - No



3. Factorial

Problem: Write a Java program to compute the factorial of a given number.

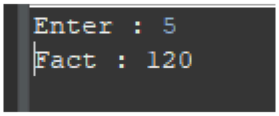
Test Cases:

Input: 5
Output: 120
Input: 0
Output: 1

Code :

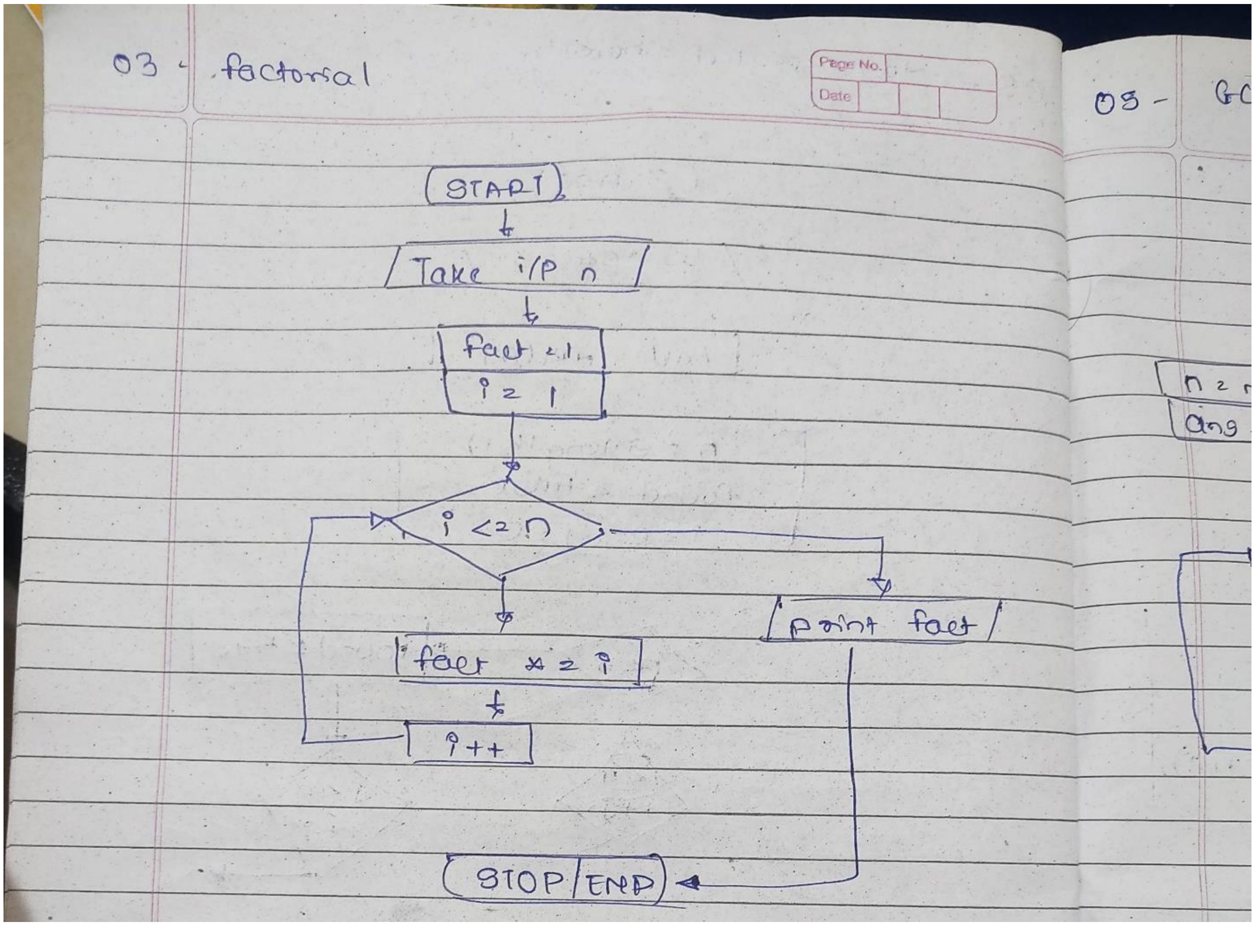
```
public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

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

        int n = sc.nextInt(), fact = 1;



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

            fact *=i;

        }

        System.out.print("Fact : "+fact);

    }
```

O/P:



```
Enter : 5
Fact : 120
```

FlowChart:


03 - factorial
Page No.
Date
START
Take i/P n
fact =1
i = 1
i <= n
fact *= i
i++
print fact
STOP/END
05 - GC
n = r
ans

4. Fibonacci Series
Problem: Write a Java program to print the first n numbers in the Fibonacci series.

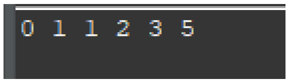
Test Cases:

Input: n = 5
Output: [0, 1, 1, 2, 3]
Input: n = 8
Output: [0, 1, 1, 2, 3, 5, 8, 13]

Code :

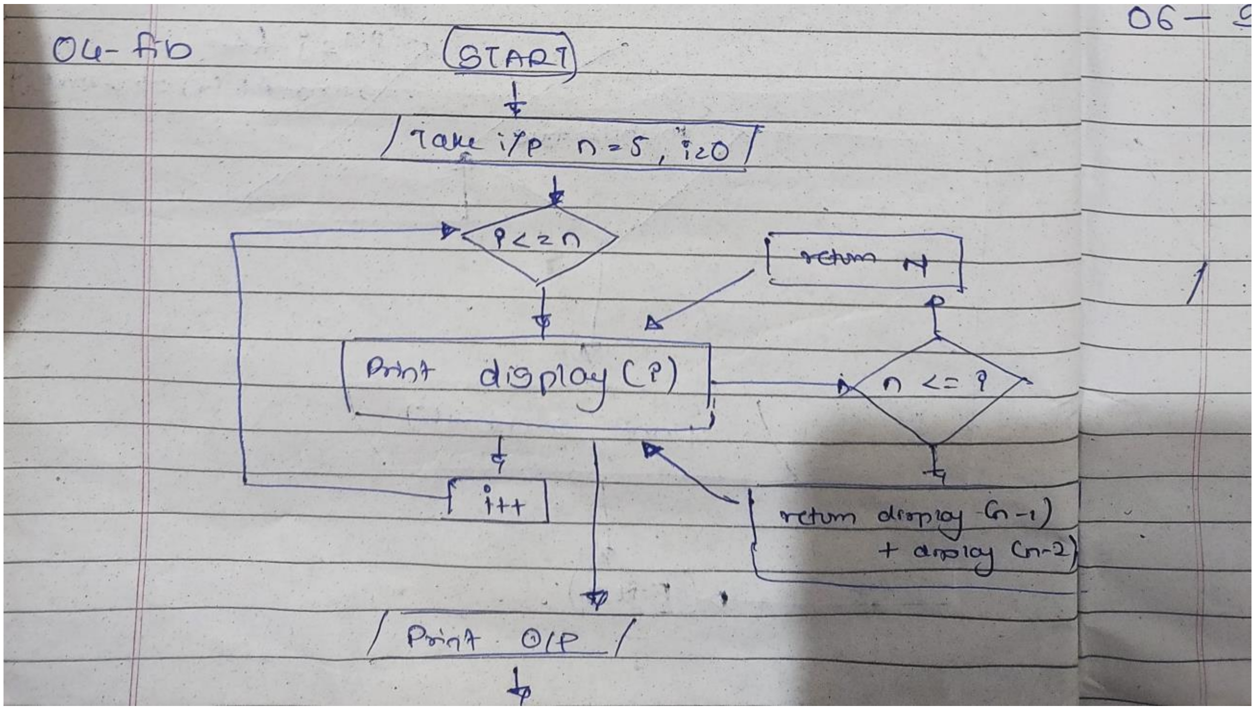
```
static int display(int n) {

                if (n <= 1) {
                        return n;
                }
                return display(n-1) + display(n-2);
        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                int n = 5;

                for (int i = 0; i <= n; i++) {
                        System.out.print(display(i)+" ");
                }

        }
```

O/P:



```
0 1 1 2 3 5
```

FlowChart :



04- fib
START
Take i/p n=5, i=0
i<=n
return n
Print display (i)
n <= i
i++
return display (n-1)
+ display (n-2)
Print O/P
06-

5. Find GCD
Problem: Write a Java program to find the Greatest Common Divisor (GCD) of two numbers.

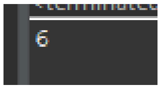
Test Cases:

Input: a = 54, b = 24
Output: 6
Input: a = 17, b = 13
Output: 1

Code :

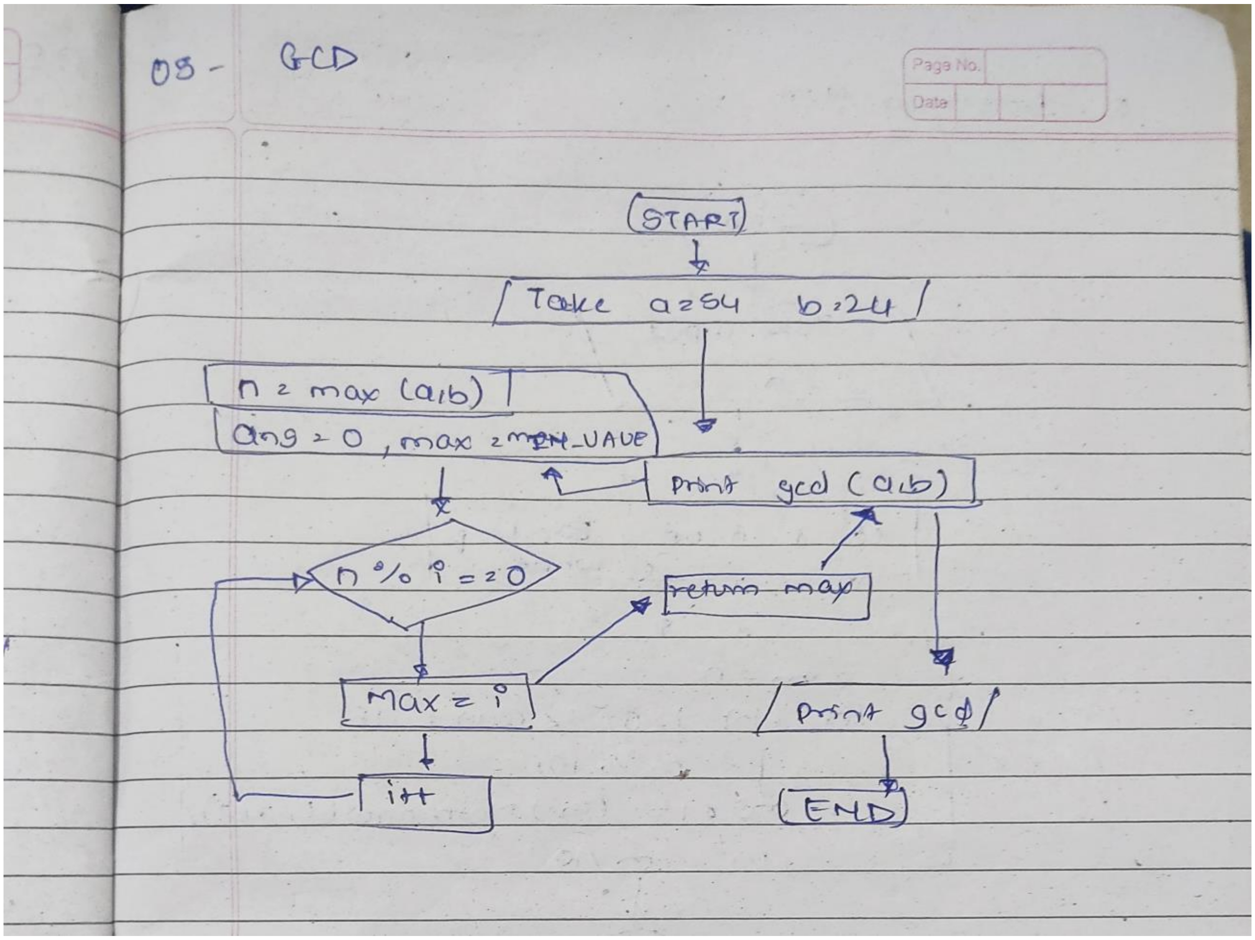
```
static int gcd(int a, int b) {

                int n = Math.max(a, b);
                int ans=0, max = Integer.MIN_VALUE;
                for (int i = 1; i <= Math.sqrt(n); i++) {
                        if (n  % i == 0) {
                                max = i;
                        }
                }
                return  max;

        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                int a = 54;
                int b = 24;

                System.out.println(gcd(a,b));;

        }
```

O/P:



```
6
```

FlowChart:


08 - GCD
Page No.
Date
START
Take a=54 b=24
n = max (a,b)
ans = 0 , max = MIN_VAUE
print gcd (a,b)
n % i == 0
return max
max = i
i++
print gcd
END

6. Find Square Root

Problem: Write a Java program to find the square root of a given number (using integer approximation).

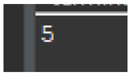
Test Cases:

Input: x = 16
Output: 4
Input: x = 27
Output: 5

Code :

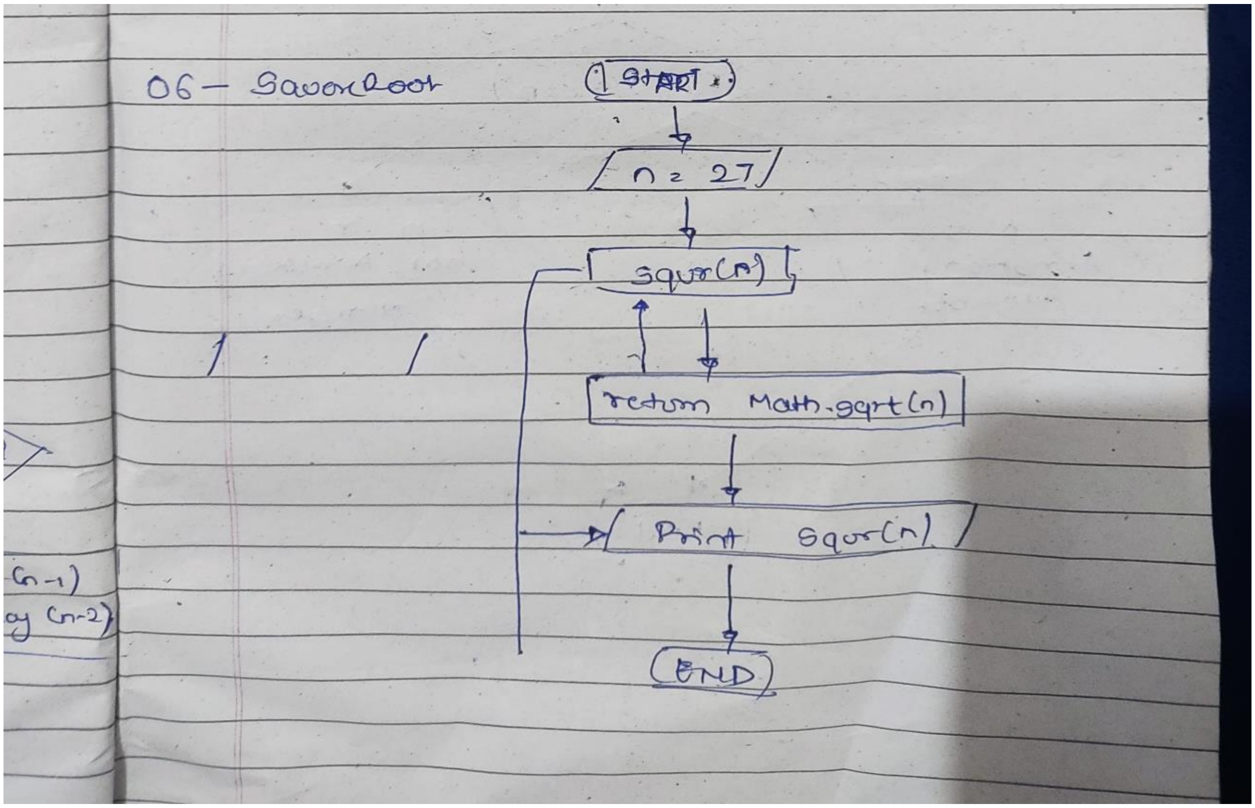
```
static int squr(int n) {
		return (int) Math.sqrt(n);
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		int n = 27;
		System.out.println(squr(n));;
	}
```

O/P:



```
5
```

FlowChart:


06— SquarRoot
START
n = 27
squr(n)
return Math.sqrt(n)
Print squr(n)
END
(n-1)
y (n-2)

7. Find Repeated Characters in a String

Problem: Write a Java program to find all repeated characters in a string.

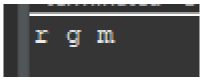
Test Cases:

Input: "programming"
Output: ['r', 'g', 'm']
Input: "hello"
Output: ['l']

Code :

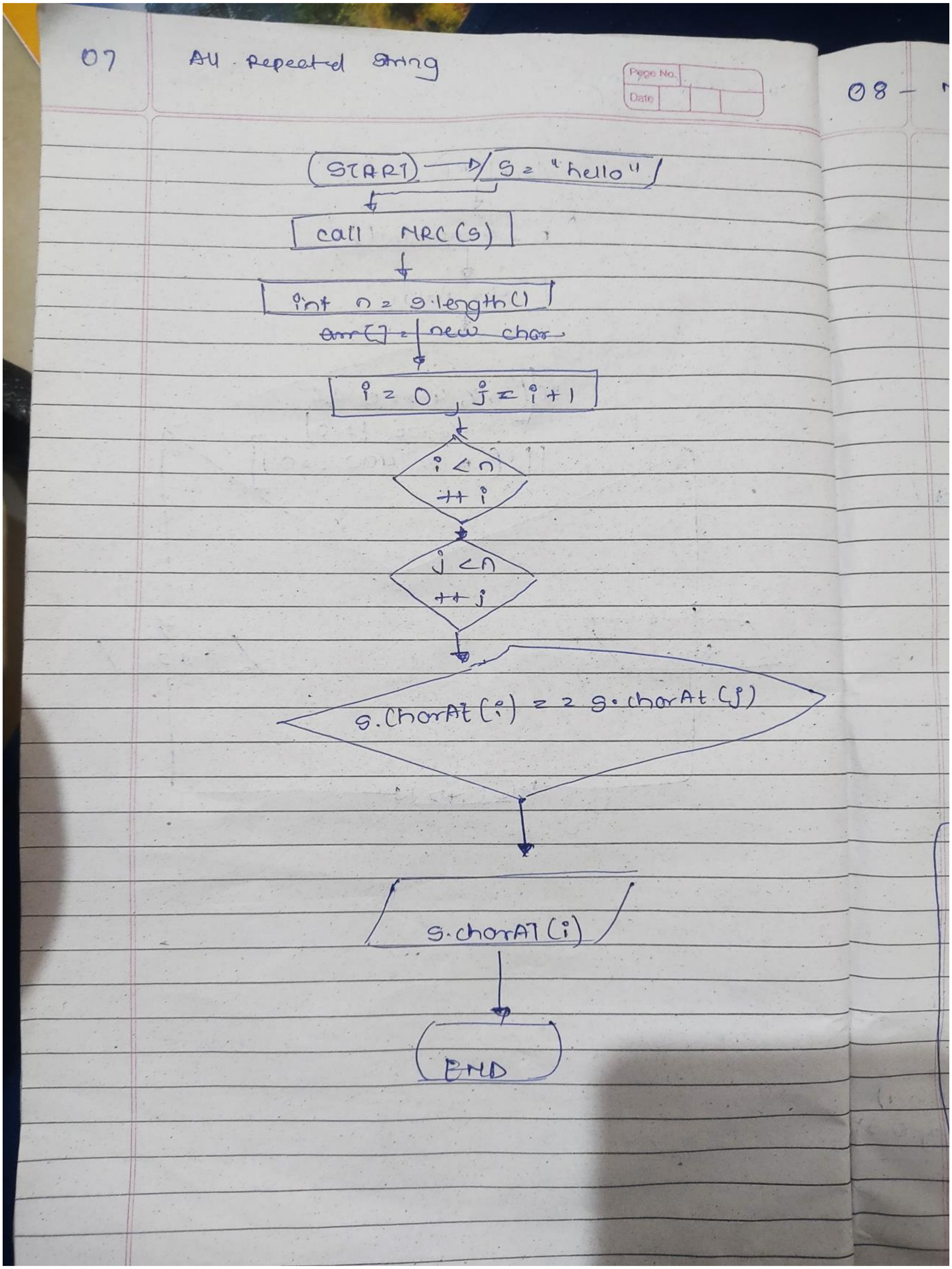
```
public static void NRC(String s)
    {
        int n = s.length();

        for (int i = 0; i < n; ++i) {
            for (int j = i+1; j < n; ++j) {
                if ( s.charAt(i) == s.charAt(j)) {

                         System.out.print(s.charAt(i)+" ");
                }
            }
        }
    }

             public static void main(String[] args) {
                         String s = "programming";
                         NRC(s);

             }
```

O/P:



```
r g m
```

FlowChart:


07
All Repeated String
Page No.
Date
08 –
START
S = "hello"
call MRC(S)
int n = s.length()
~~arr[] = new char~~
i = 0 , j = i+1
i < n
++ i
j < n
++ j
s.charAt(i) = = s.charAt(j)
s.charAT(i)
END

8. First Non-Repeated Character
Problem: Write a Java program to find the first non-repeated character in a string.

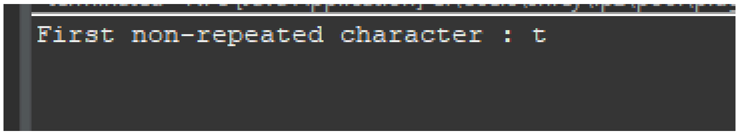
Test Cases:

Input: "stress"
Output: 't'
Input: "aabbcc"
Output: null

Code :

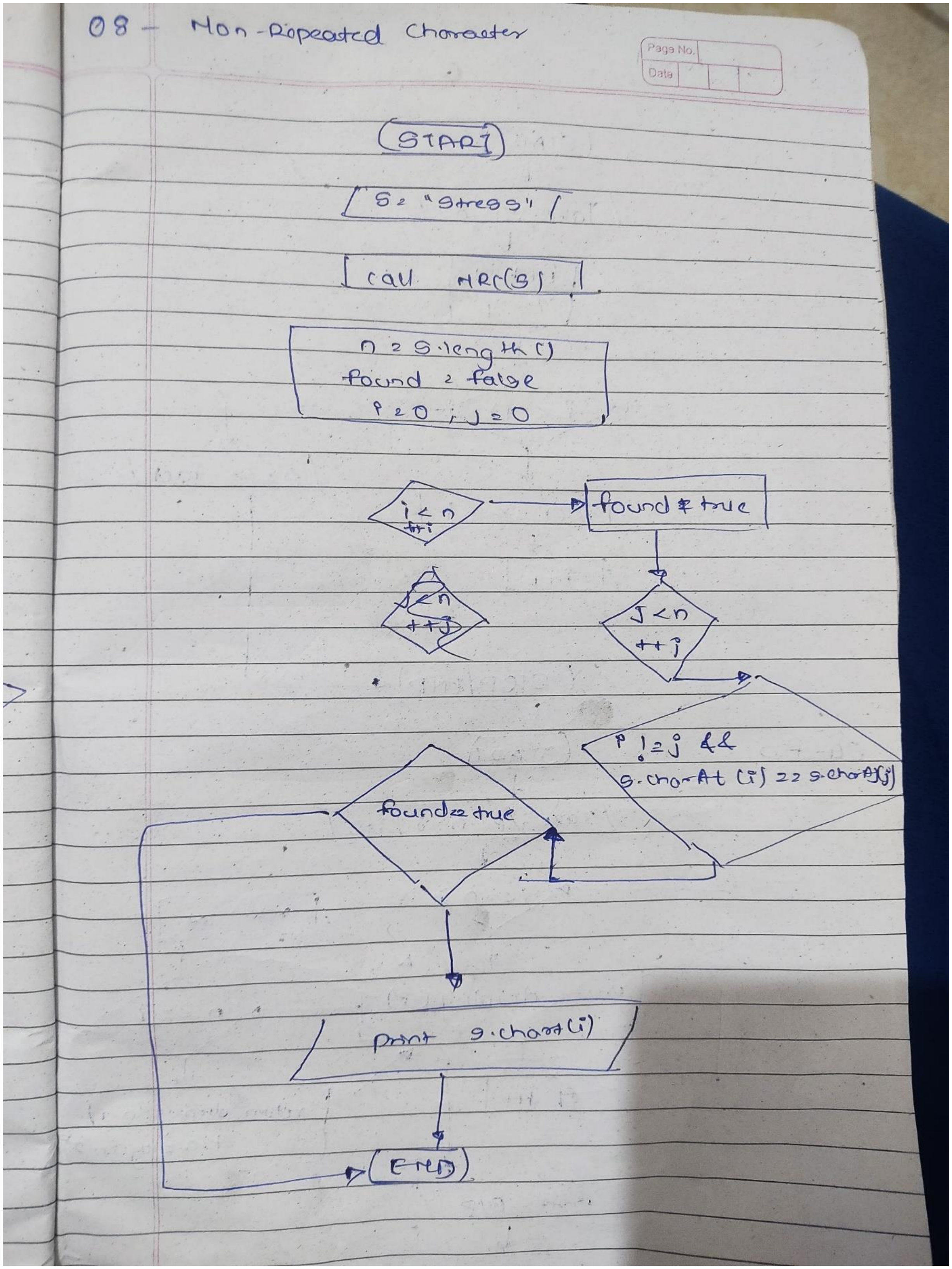
```
public static void NRC(String s) {
        int n = s.length();
        boolean found = false;

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

            for (int j = 0; j < n; ++j) {
                if (i != j && s.charAt(i) == s.charAt(j)) {
                    found = false;
                    break;
                }
            }

            if (found) {
                System.out.println("First non-repeated character : " + s.charAt(i));
                return;
            }
        }

        System.out.println("No non-repeated character found.");
    }

             public static void main(String[] args) {
                          // TODO Auto-generated method stub
                          String s = "stress";
                          NRC(s);

             }
```

O/P:



```
First non-repeated character : t
```

FlowChart:


08 - Non-Repeated Character
Page No.
Date
START
S = "stress"
call NRC(S)
n = s.length()
found = false
i = 0 ; j = 0
i < n
++i
found = true
j < n
++j
j < n
++j
i != j &&
s.charAt (i) == s.charAt(j)
found == true
print s.charAt(i)
END

9. Integer Palindrome
Problem: Write a Java program to check if a given integer is a palindrome.

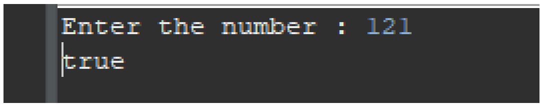
Test Cases:

Input: 121
Output: true
Input: -121
Output: false

Code :

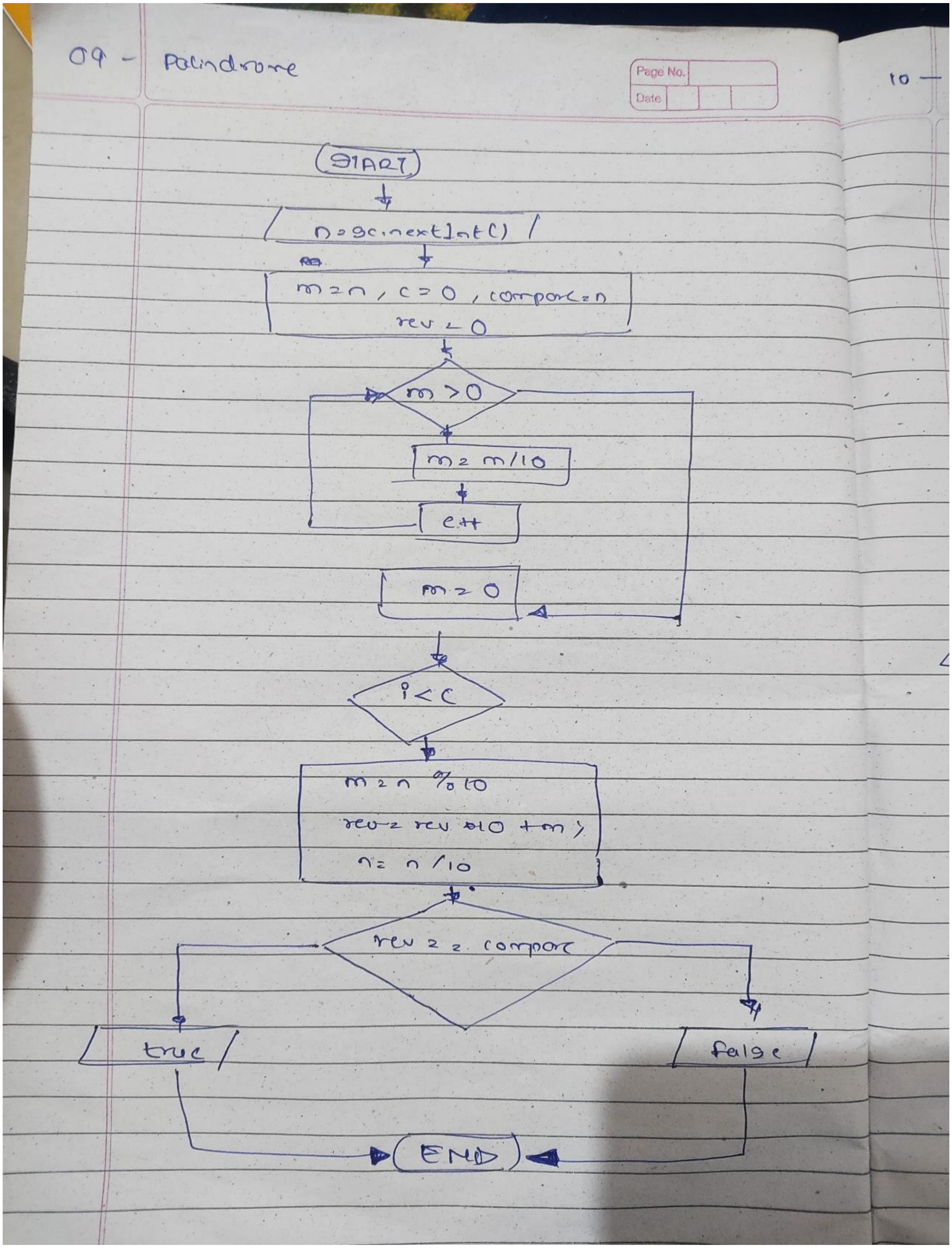
```
public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number : ");
        int n = sc.nextInt();
        int m=n,c=0, compare = n;

        while (m > 0) {
            m = m/10;
            c++;
        }
        m = 0;
        int rev =0;

        for (int i = 0; i < c; i++) {
            m = n % 10;
            rev = rev * 10 + m ;
            n = n / 10;

        }

        if (rev == compare ) {
            System.out.println("true");
        }
        else {
            System.out.println("false");
        }
    }
```

O/P:



```
Enter the number : 121
true
```

FlowChart:


09 - Palindrome
Page No.
Date
10 —
START
n=sc.nextInt()
m=n, c=0, compare=n
rev = 0
m>0
m= m/10
c++
m= 0
i<c
m = n % 10
rev= rev *10 + m;
n= n /10
rev = = compare
true
false
END

10. Leap Year
Problem: Write a Java program to check if a given year is a leap year.

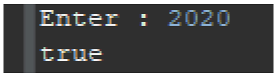
Test Cases:

Input: 2020
Output: true
Input: 1900
Output: false

Code:

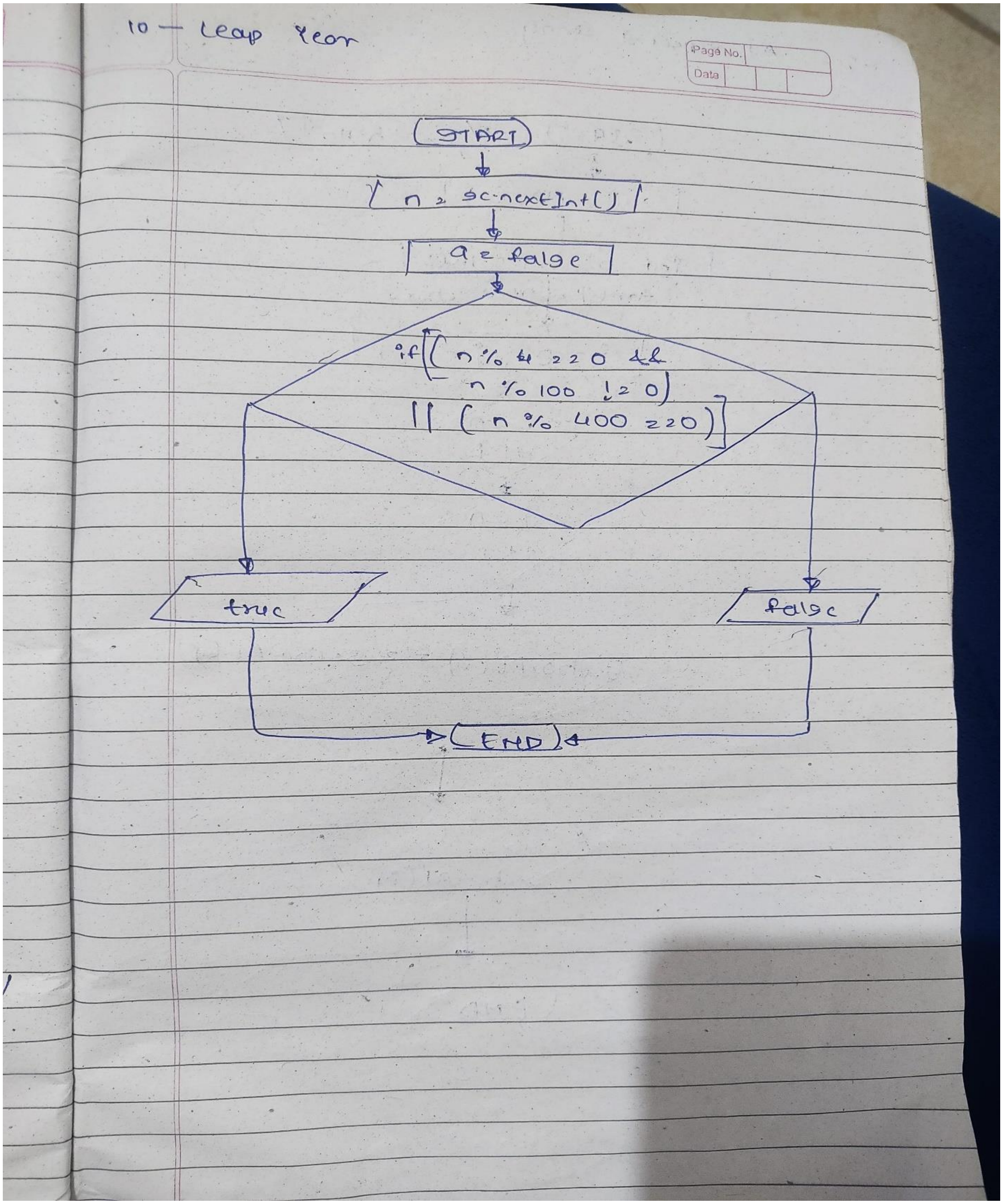
```
public static void main(String[] args) {

                // TODO Auto-generated method stub

                Scanner sc = new Scanner(System.in);

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

                int n = sc.nextInt();

                boolean a = false ;


                if (( n % 4 == 0 && n % 100 != 0) || (n % 400 == 0)){

                        a = true;

                }


                System.out.println(a);


        }
```

O/P:



```
Enter : 2020
true
```

FlowChart:


10 — Leap Year
Page No.
Data
START
n = sc.nextInt()
a = false
if ((n % 4 == 0 && n % 100 != 0) || (n % 400 == 0))
true
false
END