Assignment no 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

-Program

```
import java.util.*;

class Q1{

        public static void main (String args[])

        {

                Scanner sc = new Scanner (System.in);

                System.out.println("Enter your number");

                int n = sc.nextInt();

                int rem;

                //int div;

                int i=n;

                int temp=0;

                while(n>0)

                {

                                rem = n%10;
```

```
                temp += rem*rem*rem;
                //System.out.println(temp);
                n = n/10;
            }
            if(temp==i)
            {
                System.out.println("it is amrstrong number");
            }
            else{
                System.out.println("it is not amrstrong nu");
                }
        }
}
```

Flowchart:

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

Program:

```
import java.util.*;
class Q2{
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
```

```
                System.out.println("Enter the number");
                int n = sc.nextInt();
                int i=2;
                boolean flag = false;
                if (n== 0 || n== 1) {
      flag = true;
    }

    while (i <= n / 2) {
      if (n % i == 0) {//non prime condition
        flag = true;
        break;
      }
     ++i;
    }
    if (!flag)
      System.out.println(n + " is a prime number.");
    else
      System.out.println(n + " is not a prime number.");
  }
}
```

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

Program:

```
import java. util.*;
class Q3{
        public static void main(String args[]){
                Scanner sc = new Scanner(System.in);
                System.out.println("Enter your number:");
                int n = sc.nextInt();
                int fact=1;
                if (n == 0)
                {
        fact = 1;  // 0! is defined as 1
    }
                else
                {
                for(int i=n; i>=1;--i)
                        {
                        fact = i*fact;
                        //System.out.println(+fact);
                        }
                }
                System.out.println(+fact);
        }
}
```

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]

```
        import java.util.*;
        class Q4{
        //static int sum = 0;
        static int fib(int a){
                if(a<=1){
                        return a;
                }
                return fib(a-1)+fib(a-2);
```

```
        }
        public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
          //int sum=fib(a);
        System.out.println(fib(a));


}
}
```

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

Program :

```
import java.util.*;
class  Q5{
        public static void main(String args[]){
                gcd(54,24);
        }
        public static void gcd(int num1,int num2){
                while(num2!=0){
                        int t = num2;
                        num2= num1%num2;
                        num1 =t;
                }
                System.out.println(num1);
        }
}
```

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

Program:

```
import java.util.*;
class sq{
public static void main(String args[]){
```

```
Scanner sc = new Scanner(System.in);
System.out.println("Enter your number;");
int n = sc.nextInt();
System.out.println("find out squre root of number:");
//int squre = Math.sqrt(n));
System.out.println(Math.sqrt(n));
}
}
```

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']
Program:

```
import java.util.*;//non repeated
class Q2{
public static void main(String args[]){
	Scanner sc = new Scanner(System.in);
	String str = sc. nextLine();
	int n =  str.length();
	System.out.println(n);
	String arr[] = new String[n];
		for (int i = 0; i < n; i++)
			{
				arr[i]=String.valueOf(str.charAt(i));
			}
		String ar[] = new String[n];
		int count=0;
		for(int i =0;i<str.length();i++)
		{
			boolean temp=true;
			for(int j=0;j<count;j++)
			{
				if(arr[i].equals(ar[j]))
					{
						temp = false;
						break;
					}
			}
			if(temp)
			{
				ar[count]=arr[i];
				count++;
			}
		}
```

```
                System.out.println("Number of unique elements: " +count);
        for(int i=0;i<count;i++)
        {
                System.out.println(ar[i]+ "");
        }


}
}
```

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

Program:

```
import java.util.*;//non repeated
class Q2{
public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        String str = sc. nextLine();
        int n = str.length();
        System.out.println(n);
        String arr[] = new String[n];
                for (int i = 0; i < n; i++)
                        {
                                arr[i]=String.valueOf(str.charAt(i));
                        }
                String ar[] = new String[n];
                int count=0;
                for(int i =0 ; i<str.length();i++)
                {
                        boolean temp=false;
                        for(int j=0;j<count;j++)
                        {
                                if(arr[i].equals(ar[j]))
                                        {
                                                temp = true;
                                                break;
                                        }
                        }
                        if(!temp)
                        {
                                ar[count]=arr[i];
                                count++;
                        }
                }
                System.out.println("Number of unique elements: " +count);
```

```
        for(int i=0;i<count;i++)
        {
                System.out.println(ar[i]);
        }


}
}
```

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

```
import java.util.*;
        class palin{
                public static void main(String args[]){
                        Scanner sc = new Scanner(System.in);
                        //int n = sc.nextInt();
                        int num = sc.nextInt();
                        int num1= num;
                        int reverse=0;
                        //System.out.println(num);
                        //int ar[] = new ar[n];
                        while(num!=0){
                        int last = num%10;
                         reverse = reverse * 10 + last;
                         //System.out.println(reverse);
                          //int temp = rem;
                          num  = num/10;
                        //System.out.println(+reverse);
                        }System.out.println(+reverse);
                        if(num1 == reverse){
                                System.out.println("it is palindram number");
                        }
                        else{
                                System.out.println("it is not palindrama");
                                }
                }
        }
```

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
Program:

```
import java.util.*;
class leap{
        public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
                int year = sc.nextInt();
                System.out.println(year);
                        if(year%400==0 || year%4==0 && year%100!=0){
                        System.out.println("year is  leap");
                        }
                else{
                        System.out.println("year is not leap");
                                        }
                                }
                        }
```