Website

https://www.curiousinfotechsolutions.com

Java Programming

Day-9

Sunny Number Series	https://youtu.be/KsOT_pbShVA
Tech Number Series	https://youtu.be/3cTjh4we9FY
Armstrong Number and Peterson Number Series	https://youtu.be/_fnGWwssE6I
Palindrome Number and Factorial Number Series	https://youtu.be/EH71Hz50C4Q
Fibonacci Series and Prime Number Series	https://youtu.be/OBrOTaL-XJo

Email id:
info@curiousinfotechsolutions.com

Syllabus Day-9

Java Programming Logic

- ✓**Exp-1:** Fibonacci Series
- ✓**Exp-2:** Prime Number Series
- ✓**Exp-3:** Palindrome Number Series
- ✓**Exp-4**: Factorial Number Series
- ✓**Exp-5:** Armstrong Number Series
- ✓**Exp-6:** Peterson Number Series
- ✓**Exp-7:** Sunny Number Series
- ✓**Exp-8:** Tech Number Series

Exp-1:
Fibonacci Series
0 1 1 2 3 5 8 13 21 34


```
class Main
{
    public static void main(String args[])
    {
        int n1=0,n2=1,n3,i,count=10;
        //printing 0 and 1
        System.out.print(n1+" "+n2);
        //loop starts from 2 because 0 and 1 are already printed
        for(i=2;i<count;++i)
        {
            n3=n1+n2;
            System.out.print(" "+n3);
            n1=n2;
            n2=n3;
        }
    }
}
```

https://www.curiousinfotechsolutions.com

Exp-2: 0 to 50 Prime Number

2, 3, 5, 7, 11, 13, 17, 19,23, 29, 31, 37, 41, 43, 47


```
public class Main{
static void checkPrime(int n)
  {
      int i,m=0,flag=0;
      m=n/2;
      if(n==0||n==1) flag=1;
      else
      {
       for(i=2;i<=m;i++)
            if(n % i==0) { flag=1; break;}
       if(flag==0) System.out.print(n+" ");
      }//end of else
  }
   public static void main(String args[])
   {
    int i;
    for (i=0; i<=50; i++)
      checkPrime(i);
   }
}
```

Exp-3:

Palindrome Program

A palindrome number is a number that is same after reverse. 151 = 151, 123 = 321

For Example:
151, 545, 34543

From 1 to 200

Palindrome Number Algorithm

- Get the number to check for palindrome a=151
- H the number in temporary variable t = a
- Reverse the number revN
- Compare the temporary number with reversed number
- If both numbers are same, print "palindrome number"
- Else print "not palindrome number"

Display 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 22, 33, 44, 55, 66, 77, 88, 99, 101, 111, 121, 131, 141, 151, 161, 171, 181, 191,


```
class Main{
 static void palindromeNumber(int n)
 {
       int r,sum=0,temp;
       //It is the number variable to be checked for palindrome
       temp=n;
       while(n>0){
         r=n%10;  //getting remainder
         sum=(sum*10)+r;
         n=n/10;
       }
       if(temp==sum)
         System.out.print(temp+", ");
 }
}public static void main(String args[]){
   int i;
   for(i=0;i<=200;i++)
      palindromeNumber(i);
   }
```

Exp-4:

Factorial Program

4! = 4*3*2*1 = 24

5! = 5*4*3*2*1 = 120

Display:

0->1
1->1
2->2
3->6
4->24
5->120
6->720
7->5040
8->40320
9->362880
10->3628800


```
class Main{
static void FactorialNumber(int N)
{
  int i,fact=1;
  for(i=1;i<=N;i++)
     fact=fact*i;
  System.out.println(fact+" ");
 }
 public static void main(String args[]){
    for(int i=0; i<=10; i++)
     FactorialNumber(i);
 }
}
```

Exp-5:
Armstrong Number

An **Armstrong** number is a positive m-digit number that is equal to the sum of the m^{th} powers of their digits.

- For Example:

1634: $1^4 + 6^4 + 3^4 + 4^4 = 1 + 1296 + 81 + 256 =$ **1643**

Display:

1, 2, 3, 4, 5, 6, 7, 8, 9, 153, 370, 371, 407

```
public class Main
{
   //function to check if the number is Armstrong or not
   static boolean isArmstrong(int n)
   {
   int temp, digits=0, last=0, sum=0;
   //assigning n into a temp variable
   temp=n;
   //loop execute until the condition becomes false
   while(temp>0) {temp = temp/10;   digits++; }
   temp = n;
  while(temp>0)
   {
   //determines the last digit from the number
   last = temp % 10;
  //calculates the power of a number up to digit times and add the resultant to the sum variable
   sum +=  (Math.pow(last, digits));
   //removes the last digit
   temp = temp/10;
   }
   //compares the sum with n  returns true if sum and n are equal else returns false if sum and n are not equal
   if(n==sum)   return true; else return false;
}
```



```
//driver code
public static void  main(String args[])
{
   for(int i=0;i<=500;i++)
   {
     if(isArmstrong(i))
      System.out.print(i+", ");
   }
}
```

Exp-6:
Peterson Number

A number is said to be Peterson if the sum of factorials of each digit is equal to the sum of the number itself.

- For Example
  - Number = 145
  - 145 = !1 + !4 + !5
  - =1+4*3*2*1+5*4*3*2*1
  - =1+24+120
  - **145=145**
- Steps to Find Peterson Number
  1. Read or initialize a number (n).
  2. Find the last digit (d) of the given number.
  3. Find the factorial (fact) of the digit.
  4. Add the factorial (fact) to a variable
  5. Repeat steps 2 to 4 until the given number becomes 0.
  6. Compare the sum with n. If both are equal, the given number is Peterson, else not.

```
public class Main {
static boolean isPeterson(int num)
{
int sum = 0;
//loop executes until the condition becomes false
while (n > 0)  {
    //determines the last digit of the given number
    int digit = n % 10;
    int fact=1;
    for(int d=1;d<=digit; d++)
        fact = fact * d;
    //determines the factorial of the digit and add it to the variable sum
    sum += fact;
    //removes the last digit of the given number
    n = n / 10;
}
//compares sum with num if they are equal returns the number itself
return (sum == num);
}
```



```
//driver code
public static void main(String args[])
{
for(int n=0;n<=50000;n++)
    if (isPeterson(n))
        System.out.println(n);
}
}
```


Exp-7:
Sunny Number

A number is called a sunny number if the number next to the given number is a perfect square.

For Example:

- Given, N=80 then N+1 will be 80+1=81, which is a perfect square of the number 9. Hence **80** is a sunny number.

Steps to Find Sunny Number

1. Read or initialize a number (**num**).
2. Add **1** to the given number i.e. **num+1**.
3. Find the **square root** of num+1.
4. If the square root is an integer, the given number is **sunny**, else not a sunny number.

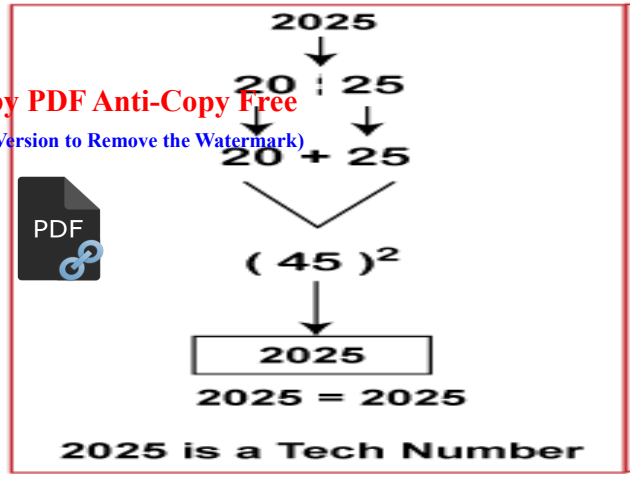
```
import java.util.*;
public class Main
{
  //function checks whether the given is a perfect square or not
static boolean findPerfectSquare(double num)
  {
  //finds the square root of the ggiven number
  double square_root = Math.sqrt(num);
  //if square root is an integer
  return((square_root - Math.floor(square_root)) == 0);
  }
```

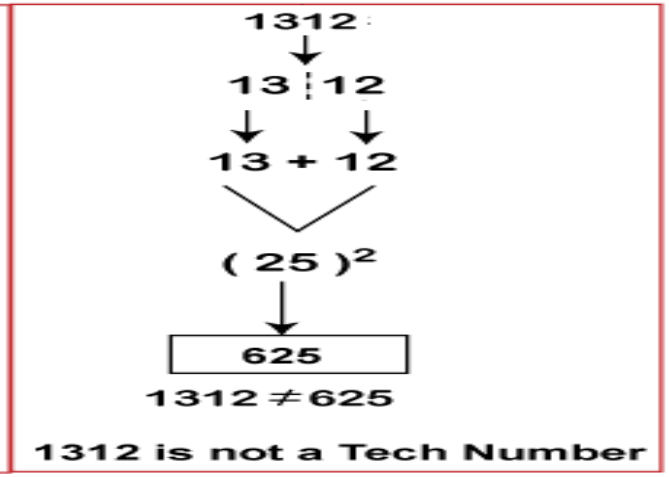


```
//function that checks whether the given number is Sunny or not
static void isSunnyNumber(int N)
  {
  //checks N+1 is perfect square or not
  if (findPerfectSquare(N + 1))
    System.out.print(N+", ");
  }
//driver code
public static void main(String args[])
  {
  for(int N=0;N<=100;N++)
    isSunnyNumber(N);
  }
}
```

Exp-8:
Tech Number

A number is called a tech number if the given number has an even number of digits, and the number can be divided exactly into two parts from the middle. After equally dividing the number, sum up the numbers and find the square of the sum. If we get the number itself as square, the given number is a tech number, else, not a tech number.





Steps to Find Tech number

- Read or initialize a number (**num**).
- Find the number of digits of the given number (**num**).
- If the number of digits is not even, the number (**num**) is not even.
- Else, split the given number into two parts (**num1** and **num2**), equally. Note that each part must contain an equal number of digits.
- Sum up the numbers (**num1+num2**) and store the result in a variable
- Find the square of the variable **sum** and store it in a variable **square**.
- Compare the **num** with the **square of the sum** if they are equal print **Tech Number,** else print **Not a Tech Number.**

For Exp:
0,
81,
2025,
3025,

```
public class Main
{
static void TechNum(int n)
{
    int num, firstHalf, secondHalf;
    int digits = 0, squareOfSum = 0;
    num = n;
    //the loop executes until the condition returns false
    while (num > 0)
    {
    //inceraments the variable digits by 1
    digits++;
    //removes the last digit of the given number
    num = num / 10;
    }
    //check if the given number has an even number of digits or not
    if (digits % 2 == 0)
    {
    num = n;
    //determines the first half of the given number
    firstHalf = num % (int) Math.pow(10, digits / 2);
    //determines the second half of the given number
    secondHalf = num / (int) Math.pow(10, digits / 2);
    //calculate the square of both the numbers
    squareOfSum = (firstHalf + secondHalf) * (firstHalf + secondHalf);
    //compares the square of the sum with the given number
    if (n == squareOfSum)
        System.out.println(n+", ");
    }
}
public static void main(String args[])
    {
    for(int i=0;i<=5000;i++)
    TechNum(i);
    }
}
```