**S P E C I A L M A T H E M A T I C A L N U M B E R S**

In nature there exist some mathematical numbers, having some recognizable, special characteristics, which can be arithmetically attributed. They can be utilized in various computations due to their inherent special attributes to arrive at a solution. Here is a list of such numbers, known by the different names, along with their mathematical characteristics and their respective coding to identify such numbers written in the High-Level Programming Language **JAVA**.

1. **ODD AND EVEN NUMBER**

A number is said to be odd, if it is not divisible by 2. Example:- 1, 3, 5, 7, 9, etc.

A number is said to be even, if it is divisible by 2. Example:- 2, 4, 6, 8, 10, etc

*Coding:*

import java.io.\*;

public class Odd\_Even

{

public static void main (String[] args) throws IOException

{

BufferedReader br = new BufferedReader (new InputStreamReader (System.in));

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

int n = Integer.parseInt (br.readLine ());

if (n%2 == 0)

{

System.out.println (n + " is an even number.");

}

else

{

System.out.println (n + " is an odd number.");

}

}

}

1. **PRIME AND COMPOSITE NUMBER**

A number is said to be prime if it is divisible by 1 and itself only and by no other number. Example:- 2, 3, 5, 7, 11, etc.

A number is said to be composite that can be divided by any positive number other than themselves. Example:- 4, 6, 8, 9, 10, etc.

*Coding:*

import java.io.\*;

public class Prime\_Composite

{

public static void main (String[] args) throws IOException

{

BufferedReader br = new BufferedReader (new InputStreamReader (System.in));

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

int n = Integer.parseInt (br.readLine ());

int f = 0;

if (n != 1)

{

for (int i = 2 ; i <= n/2 ; i++)

{

if (n % i == 0) f = 1;

}

if (f == 0)

{

System.out.println (n + " is a prime number.");

}

else

{

System.out.println (n + " is a composite number.");

}

}

else

{

System.out.println (n + " is an unit number.");

}

}

}

1. **ABUNDANT AND DEFICIENT NUMBER**

A number is said to be abundant if it is greater than the sum of its factors including itself. Example:- 12, 18, 20, 24, 30, etc.

A number is said to be deficient if it is less than the sum of its factors including itself. Example:- 1, 2, 3, 4, 5, etc.

*Coding:*

import java.io.\*;

public class Abundant\_Deficient

{

public static void main (String[] args) throws IOException

{

BufferedReader br = new BufferedReader (new InputStreamReader (System.in));

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

int n = Integer.parseInt (br.readLine ());

int sum = 0;

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

{

if (n % i == 0) sum += i;

}

if (sum < (2\*n))

{

System.out.println (n + " is a deficient number.");

}

else if (sum > (2\*n))

{

System.out.println (n + " is an abundant number.");

}

else

{

System.out.println (n + " is neither deficient nor abundant number.");

}

}

}

1. **PERFECT NUMBER**

A number is said to be perfect if it is equal to the sum of its factors excluding itself. Example:- 6, 28, 496, etc.

*Coding:*

import java.io.\*;

public class Perfect

{

public static void main (String[] args) throws IOException

{

BufferedReader br = new BufferedReader (new InputStreamReader (System.in));

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

int n = Integer.parseInt (br.readLine ());

int sum = 0;

for (int i = 1 ; i <= n/2 ; i++)

{

if (n % i == 0) sum += i;

}

if (sum == n)

{

System.out.println (n + " is a perfect number.");

}

else

{

System.out.println (n + " is not a perfect number.");

}

}

}

1. **KRISHNAMURTHY/ SPECIAL NUMBER**

A number is called a Krishnamurthy/ special number if it is equal to the sum of the factorial of its individual digits. Example:- 145

145 = 1! + 4! + 5!

*Coding:*

import java.io.\*;

public class Special

{

public static void main (String[] args) throws IOException

{

BufferedReader br = new BufferedReader (new InputStreamReader (System.in));

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

int n = Integer.parseInt (br.readLine ());

int m = n;

int factorial = 1;

int sum = 0;

while (n > 0)

{

int r = n % 10;

for (int i = 1 ; i <= r ; i++)

{

factorial \*= i;

}

sum += factorial;

factorial = 1;

n /= 10;

}

if (sum == m)

{

System.out.println (m + " is a Krishnamurthy/ Special number.");

}

else

{

System.out.println (m + " is not a Krishnamurthy/ Special number.");

}

}

}

1. **MAGIC NUMBER**

A number is said to be magic if the ultimate sum of its digits equals to 1. Example:- 352

352 = 3 + 5 + 2 = 10; 10 = 1 + 0 = 1

*Coding:*

import java.io.\*;

public class Magic

{

public static void main (String[] args) throws IOException

{

BufferedReader br = new BufferedReader (new InputStreamReader (System.in));

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

int n = Integer.parseInt (br.readLine ());

int m = n;

int num = n;

while (m > 9)

{

n = m;

m = 0;

while (n > 0)

{

int r = n % 10;

m += r;

n /= 10;

}

}

if (m == 1)

{

System.out.println (num + " is a magic number.");

}

else

{

System.out.println (num + " is not a magic number.");

}

}

}

1. **PALINDROME NUMBER**

A number is said to be palindrome if it reads the same from either end. Example:- 121, 2442, etc.

*Coding:*

import java.io.\*;

public class Palindrome

{

public static void main (String[] args) throws IOException

{

BufferedReader br = new BufferedReader (new InputStreamReader (System.in));

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

int n = Integer.parseInt (br.readLine ());

int m = n;

int reverse = 0;

int sign = (n < 0) ? -1:1;

n \*= sign;

while (n != 0)

{

int r = n % 10;

reverse = reverse \* 10 + r;

n /= 10;

}

if (reverse\*sign == m)

{

System.out.println (m + " is a palindrome number.");

}

else

{

System.out.println (m + " is not a palindrome number.");

}

}

}

1. **ARMSTRONG/ NARCISSTIC NUMBER**

A number is said to be Armstrong/ narcisstic if it is equal to the sum of the cubes of its digits. Example:- 153

153 = 13 + 53 + 33

*Coding:*

import java.io.\*;

public class Armstrong

{

public static void main (String[] args) throws IOException

{

BufferedReader br = new BufferedReader (new InputStreamReader (System.in));

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

int n = Integer.parseInt (br.readLine ());

int m = n;

int sum = 0;

while (n > 0)

{

int r = n % 10;

sum += (r\*r\*r);

n /= 10;

}

if (sum == m)

{

System.out.println (m + " is an Armstrong/ narcisstic number.");

}

else

{

System.out.println (m + " is not an Armstrong/ narcisstic number.");

}

}

}

1. **AUTOMORPHIC NUMBER**

A number is said to be automorphic if it is contained in the right-most digits of its square. Example:- 5, 6, etc.

52 = 25

62 = 36

*Coding:*

import java.io.\*;

public class Automorphic

{

public static void main (String[] args) throws IOException

{

BufferedReader br = new BufferedReader (new InputStreamReader (System.in));

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

int n = Integer.parseInt (br.readLine ());

int m = n;

int count = 0;

int l = (m\*m);

while (n > 0)

{

n /= 10;

count++;

}

double k = 1 % Math.pow (10, count);

if (k == m)

{

System.out.println (m + " is an automorphic number.");

}

else

{

System.out.println (m + " is not an automorphic number.");

}

}

}

1. **KAPREKAR/ NEON NUMBER**

A number is said to be Kaprekar/ Neon if it is equal to the sum of the numbers comprising the rightmost n digits and the leftmost n or (n-1) digits of its square. Example:- 9, 297, etc.

92 = 81; 8 + 1 = 9

2972 = 88209; 88 + 209 = 297

*Coding:*

import java.io.\*;

public class Kaprekar

{

public static void main (String[] args) throws IOException

{

BufferedReader br = new BufferedReader (new InputStreamReader (System.in));

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

int n = Integer.parseInt (br.readLine ());

int m = n;

int count = 0;

int l = (n\*n);

while (n > 0)

{

n /= 10;

count++;

}

double k = 1 % Math.pow (10, count);

double a = 1 - k;

double b = a / Math.pow (10, count);

double add = k + b;

if (add == m)

{

System.out.println (m + " is a Kaprekar/ neon number.");

}

else

{

System.out.println (m + " is not a Kaprekar/ neon number.");

}

}

}

1. **DISARIUM NUMBER**

A number is called a Disarium number if it is equal to the sum of its digits powered with their respective positions. Example:- 135

135 = 11 + 32 + 53

*Coding:*

import java.io.\*;

public class Disarium

{

public static void main (String[] args) throws IOException

{

BufferedReader br = new BufferedReader (new InputStreamReader (System.in));

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

int n = Integer.parseInt (br.readLine ());

String s = Integer.toString (n);

int m = n;

int i = s.length ();

int new\_n = 0;

while (n > 0)

{

int r = n % 10;

new\_n += Math.pow (r,i);

n /= 10;

i--;

}

if (new\_n == m)

{

System.out.println (m + " is a Disarium number.");

}

else

{

System.out.println (m + " is not a Disarium number.");

}

}

}

1. **BUZZ NUMBER**

A number is said to be Buzz if it is either divisible by 7 or leaves a remainder of 7 when divided by 10. Example:- 21, 77, etc.

*Coding:*

import java.io.\*;

public class Buzz

{

public static void main (String[] args) throws IOException

{

BufferedReader br = new BufferedReader (new InputStreamReader (System.in));

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

int n = Integer.parseInt (br.readLine ());

if (n % 7 ==0 || n % 10 ==7)

{

System.out.println (n + " is a Buzz number.");

}

else

{

System.out.println (n + " is not a Buzz number.");

}

}

}

1. **MERSENNE NUMBER**

A number is said to be Mersenne if it contains all bits in base 2 representation, i.e., in the form of 2n – 1, where n is the number of bits. Example:- 1, 3, 7, 15, 31, 63, 127 etc.

*Coding:*

import java.io.\*;

public class Mersenne

{

public static void main (String[] args) throws IOException

{

BufferedReader br = new BufferedReader (new InputStreamReader (System.in));

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

int n = Integer.parseInt (br.readLine ());

int x = n + 1;

int count = 0;

for (int i = 1 ; i <= x/2 ; i++)

{

if (x % i == 0) count++;

}

if (x % 2 == 0)

{

if (Math.pow (2, count) - 1 == n)

{

System.out.println (n + " is a Mersenne number.");

}

else

{

System.out.println (n + " is not a Mersenne number.");

}

}

else

{

System.out.println (n + " is not a Mersenne number.");

}

}

}

1. **SMITH NUMBER**

A number is called a Smith Number if the sum of its digits equals the sum of the digits of its prime factors. Example:- 666

Sum of digits: 18

Prime factors of 666: 2, 3, 3, 37

Sum of the digits of each prime factor of 666 = 2 + 3 + 3 + (3 + 7) = 18

So, 666 is a Smith Number

*Coding:*

import java.io.\*;

public class Smith

{

public boolean isPrime (int n)

{

int f = 0;

for (int i = 2 ; i <= n/2 ; i++)

{

if (n % i == 0)

{

f = 1;

}

}

if (f == 0)

{

return true;

}

else

{

return false;

}

}

public int sumOfDigits (int n)

{

int sum = 0;

while (n > 0)

{

int r = n % 10;

sum += r;

n /= 10;

}

return sum;

}

public static void main (String[] args) throws IOException

{

BufferedReader br = new BufferedReader (new InputStreamReader (System.in));

Smith obj = new Smith ();

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

int n = Integer.parseInt (br.readLine ());

int m = n;

int sumD = obj.sumOfDigits (n);

int sumF = 0;

for (int i = 2 ; i <= n ; i++)

{

if (obj.isPrime (i))

{

while (n % i == 0)

{

sumF += obj.sumOfDigits (i);

n /= i;

}

}

}

if (sumD == sumF)

{

System.out.println (m + " is a Smith number.");

}

else

{

System.out.println (m + " is not a Smith number.");

}

}

}

1. **TWISTED PRIME NUMBER**

A prime number is said to be Twisted Prime if the new number obtained after reversing the digits is also a prime number. Example:- 167

*Coding:*

import java.io.\*;

public class TwistedPrime

{

public boolean isPrime (int n)

{

int f = 0;

for (int i = 2 ; i <= n/2 ; i++)

{

if (n % i == 0)

{

f = 1;

}

}

if (f == 0)

{

return true;

}

else

{

return false;

}

}

public static void main (String[] args) throws IOException

{

BufferedReader br = new BufferedReader (new InputStreamReader (System.in));

TwistedPrime obj = new TwistedPrime ();

int n = 0;

do

{

System.out.println ("Enter a prime number: ");

n = Integer.parseInt (br.readLine ());

} while (!obj.isPrime (n) && System.out.printf (n + " is not a prime number.") != null);

int m = n;

int rev = 0 ;

while (n > 0)

{

int r = n % 10;

rev \*= (10 + r);

n /= 10;

}

if (obj.isPrime (rev))

{

System.out.println (m + " is a Twisted Prime number.");

}

else

{

System.out.println (m + " is not a Twisted Prime number.");

}

}

}

**OUTPUT OF THE PROGRAMS**

|  |  |
| --- | --- |
| **Fig. 1:** Odd and Even Number | **Fig. 2:** Prime and Composite Number |
| **Fig. 3:** Abundant and Deficient Number | **Fig. 4:** Perfect Number |
| **Fig. 5:** Krishnamurthy/ Special Number | **Fig. 6:** Magic Number |
| **Fig. 7:** Palindrome Number | **Fig. 8:** Armstrong/ Narcisstic Number |
| **Fig. 9:** Automorphic Number | **Fig. 10:** Kaprekar/ Neon Number |
| **Fig. 11:** Disarium Number | **Fig. 12:** Buzz Number |
| **Fig. 13:** Mersenne Number | **Fig. 14:** Smith Number |
| **Fig. 15:** Twisted Prime Number | |