# PROGRAMMING QUESTIONS ON NUMBER LOGIC

1. Write a program to the sum of n natural numbers.
2. Write a program to compute 1/n!.
3. Write a program to compute x^n/n!.
4. Write a program to count the number of digits in an integer.
5. Write a program to sum all digits of a number.
6. Write a program to reverse a given number.
7. Write a program to calculate G.C.D or HCF of two numbers.
8. Write a program to calculate the LCM of two numbers.
9. Write a program to find the factorial of a number.
10. Write a Program to calculate prime Factors of number.
11. Write a program to check the given number is a palindrome or not.
12. Write a program to check whether a given number is prime or not.
13. Write a program to check whether a given number is a perfect square number or not.
14. Write a program to check whether a given number is an Armstrong number or not.
15. Write a program to check whether a given number is a strong number or

not.

16. Write a program to check whether a given number is a perfect number or not.

17. Write a program to check whether a given number is a Harshad number or not.

18. Write a program to check whether a given number is an Abundant number or not.

19. Write a program to check whether a given number is an Automorphic number or not.

20. Write a Program to check whether the number is Magic Number or Not.

21. Write a program to check whether a given number is Friendly pair or not.

22. Write a Program to check whether the number is Neon Number or Not.

23. Write a Program to check whether the number is Spy Number or Not.

24. Write a Program to check whether the number is Happy Number or Not.

25. Write a Program to check whether the number is Sunny Number or Not.

26. Write a Program to check whether the number is Disarium Number or Not.

27. Write a Program to check whether the number is a Pronic Number or Not.

28. Write a Program to check whether the number is a Trimorphic Number or Not.

29. Write a Program to check whether the number is an Evil Number or Not.

30. Write a program to find out all palindrome numbers present within a given range.

31. Write a program to find out all primes numbers present within a given range.

32. Write a program to find out all perfect square numbers present within a given range.

33. Write a program to find out all Armstrong numbers present within a given range.

34. Write a program to find out all Strong numbers present within a given range.

35. Write a program to find out all Perfect numbers present within a given range.

36. Write a program to find out all Harshad numbers present within a given range.

37. Write a program to find out all Abundant numbers present within a given range.

38. Write a program to find out all Automorphic numbers present within a given range.

39. Write a Program to Find out all Magic numbers present within a given range.

40. Write a Program to Find out all Neon numbers present within a given range.

41. Write a Program to Find out all Spy numbers present within a given range.

42. Write a Program to Find out all Happy numbers present within a given

range.

43. Write a Program to Find out all Sunny numbers present within a given range.

44. Write a Program to Find out all the Disarium numbers present within a given range.

45. Write a Program to Find out all Pronic numbers present within a given range.

46. Write a Program to Find out all Trimorphic numbers present within a given range.

47. Write a Program to Find out all Evil numbers present within a given range.

48. Write a program to Find the nth palindrome number.

49. Write a program to find the nth prime number.

50. Write a Program to Find nth Perfect Square Number.

51. Write a Program to Find nth Armstrong Number.

52. Write a program to find the nth strong number.

53. Write a program to find the nth perfect number.

54. Write a program to find the nth Hashed number.

55. Write a Program to Find nth Abundant Number.

56. Write a program to find the nth automorphic number.

57. Write a Program to find the nth Magic Number.

58. Write a Program to Find nth Neon Number.

59. Write a Program to Find nth Spy Number.

60. Write a Program to Find nth Happy Number.

61. Write a Program to Find nth Sunny Number.

62. Write a Program to Find nth Disarium Number.

63. Write a Program to Find nth Pronic Number.

64. Write a Program to Find nth Trimorphic Number.

65. Write a Program to Find nth Evil Number.

66. Write a program to Calculate Reverse a number using Recursion.

67. Write a program to find the Generic root of a number.

68. Write a program to find out how many 1 and 0 in a given number.

69. Write a program to add between 2 numbers without using arithmetic operators.

70. Write a program to find the largest digit in a number.

71. Write a program to find the smallest digit in a number.

72. Write a program to calculate Amicable pairs.

73. Write a program to find the 2nd largest digit in a given number.

74. Write a program to find the 2nd smallest digit in a given number.

75. Write a program to find the number of odd and even digits in the given number.

---

**76** Write a program to check whether a given number is an ugly number.
In number system, ugly numbers are positive numbers whose only prime factors are 2, 3 or 5. First 10 ugly numbers are 1, 2, 3, 4, 5, 6, 8, 9, 10, 12. By convention, 1 is included.

**77** Write a Java program to classify Abundant, deficient and perfect number (integers) between 1 to 10,000.
In number theory, an abundant number is a number for which the sum of its proper divisors is greater than the number itself.

Example :
The first few abundant numbers are:
12, 18, 20, 24, 30, 36, 40, 42, 48, 54, 56, 60, 66, 70, 72, 78, 80, 84, 88, 90, 96, 100, 102,…
The integer 12 is the first abundant number. Its proper divisors are 1, 2, 3, 4 and 6 for a total of 16.
Deficient number: In number theory, a deficient number is a number n for which the sum of divisors $\sigma(n)<2n$, or, equivalently, the sum of proper divisors (or aliquot sum) $s(n)<n$. The value $2n - \sigma(n)$ (or $n - s(n)$) is called the number's deficiency.
As an example, divisors of 21 are 1, 3 and 7, and their sum is 11. Because 11 is less than 21, the number 21 is deficient. Its deficiency is $2 \times 21 - 32 = 10$.
The first few deficient numbers are:
1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 19, 21, 22, 23, 25, 26, 27, 29, 31, 32, 33, …….
Perfect number: In number system, a perfect number is a positive integer that is equal to the sum of its proper positive divisors, that is, the sum of its positive divisors excluding the number itself.
Equivalently, a perfect number is a number that is half the sum of all of its positive divisors (including itself) i.e. $\sigma 1(n) = 2n$.
The first perfect number is 6. Its proper divisors are 1, 2, and 3, and 1 + 2 + 3 = 6. Equivalently, the number 6 is equal to half the sum of all its positive divisors: ( 1 + 2 + 3 + 6 ) / 2 = 6. The next perfect number is 28 = 1 + 2 + 4 + 7 + 14. This is followed by the perfect numbers 496 and 8128.
*Expected Output* :
Number Counting [(integers) between 1 to 10,000]:
Deficient number: 7508
Perfect number: 4
Abundant number: 2488

**78** Write a program to generate random integers in a specific range.

**79** Write a program to generate and show all Kaprekar numbers less than 1000.
*Expected Output* :

```
1       1         0 + 1
9       81        8 + 1
45      2025      20 + 25
55      3025      30 + 25
99      9801      98 + 01
297     88209     88 + 209
703     494209    494 + 209
999     998001    998 + 001
8 Kaprekar numbers.
```

**80 W**rite a Java program to find the number of seed Lychrel number candidates and related numbers for n in the range 1..10000 inclusive. (With that iteration limit of 500).

A Lychrel number is a natural number that cannot form a palindrome through the iterative process of repeatedly reversing its digits and adding the resulting numbers. This process is sometimes called the 196-algorithm, after the most famous number associated with the process.

The first few Lychrel numbers are 196, 295, 394, 493, 592, 689, 691, 788, 790, 879, 887, ... .

*Expected Output* :

5 Lychrel seeds: [196, 879, 1997, 7059, 9999]

244 Lychrel related

5 Lychrel palindromes: [196, 879, 1997, 7059, 9999]

**81** Write a program to generate and show the first 15 narcissistic decimal numbers.

*Expected Output* :

```
0 1 2 3 4 5 6 7 8 9 153 370 371 407 1634
```

**82** Write a Java program to display first 10 lucus numbers.

The Lucas numbers or series are an integer sequence named after the mathematician François Édouard Anatole Lucas, who studied both that sequence and the closely related Fibonacci numbers. Lucas numbers and Fibonacci numbers form complementary instances of Lucas sequences.

The sequence of Lucas numbers is: 2, 1, 3, 4, 7, 11, 18, 29, ….

*Expected Output* :

```
First ten Lucas a numbers:
```

```
2
1
3
4
7
11
18
29
47
76
```

**83** Write a program to print out the first 10 Catalan numbers by extracting them from Pascal's triangle.

In combinatorial mathematics, the Catalan numbers form a sequence of natural numbers that occur in various counting problems, often involving recursively-defined objects. They are named after the Belgian mathematician Eugène Charles Catalan.

The first Catalan numbers for n = 0, 1, 2, 3, … are 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452,

```
List 10 Catalan numbers:-

1
2
5
14
42
132
429
1430
4862
16796
```

**84** Write a program to find and print the first 10 happy numbers.

Happy number: Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number equals 1, or it loops endlessly in a cycle which does not include 1.

Example: 19 is a happy number

$1^2 + 9^2=82$

$8^2 + 2^2=68$

$6^2 + 8^2=100$

$1^2 + 0^2 + 0^2=1$

*Expected Output*

```
First 10 Happy numbers:
1
7
10
13
19
23
28
31
```

**85** Write a program to check whether a given number is a happy number or unhappy number.
Happy number: Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number equals 1, or it loops endlessly in a cycle which does not include 1.
An unhappy number is a number that is not happy.
The first few unhappy numbers are 2, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15, 16, 17, 18, 20.

*Expected Output*

```
Input a number: 5
Unhappy Number
```

**86** Write a program to check whether a given number is a Disarium number or unhappy number.
A Disarium number is a number defined by the following process :
Sum of its digits powered with their respective position is equal to the original number.
For example 175 is a Disarium number :
As $1^1+3^2+5^3 = 135$
Some other DISARIUM are 89, 175, 518 etc.
A number will be called Disarium if the sum of its digits powered with their respective position is equal with the number itself. Sample Input: 135.

*Expected Output*

```
Input a number : 25
Not a Disarium Number.
```

**87 .** Write a program to check whether a number is a Harshad Number or not.
In recreational mathematics, a harshad number in a given number base, is an integer that is divisible by the sum of its digits when written in that base.
Example: Number 200 is a Harshad Number because the sum of digits 2 and 0 and 0 is 2(2+0+0) and 200 is divisible by 2. Number 171 is a Harshad Number because the sum of digits 1 and 7 and 1 is 9(1+7+1) and 171 is divisible by 9.
*Expected Output*

```
Input a number : 353

353 is not a Harshad Number.
```

**88** Write a program to check whether a number is a Pronic Number or Heteromecic Number or not.
A pronic number is a number which is the product of two consecutive integers, that is, a number of the form n(n + 1).
The first few pronic numbers are:
0, 2, 6, 12, 20, 30, 42, 56, 72, 90, 110, 132, 156, 182, 210, 240, 272, 306, 342, 380, 420, 462 … etc.

*Expected Output*
Input a number : 110
Pronic Number.

**89 .** Write a program to check whether a number is a Duck Number or not.
Note: A Duck number is a number which has zeroes present in it, but there should be no zero present in the beginning of the number. For example 3210, 7056, 8430709 are all duck numbers whereas 08237, 04309 are not.
*Expected Output*

```
Input a number : 3210
Duck number
```

**90 Wr**ite a program to check two numbers are Amicable numbers or not.
Amicable numbers are two different numbers so related that the sum of the proper divisors of each is equal to the other number.
The first ten amicable pairs are: (220, 284), (1184, 1210), (2620, 2924), (5020, 5564), (6232, 6368), (10744, 10856), (12285, 14595), (17296, 18416), (63020, 76084), and (66928, 66992).
*Expected Output*

```
Input the first number: 220
Input the second number: 284
These numbers are amicable.
```

**91** Write a program to check if a given number is circular prime or not.
Circular Prime : A circular prime is a prime number with the property that the number generated at each intermediate step when cyclically permuting its (base 10) digits will be prime.
For example, 1193 is a circular prime, since 1931, 9311 and 3119 all are also prime. A circular prime with at least two digits can only consist of combinations of the digits 1, 3, 7 or 9, because having 0, 2, 4, 6 or 8 as the last digit makes the number divisible by 2, and having 0 or 5 as the last digit makes it divisible by 5.
Input Data:
Input a number: 35
*Expected Output*

```
It is not a Circular Prime number.
```

92 write a program to check a number is a cube or not.
In arithmetic and algebra, the cube of a number n is its third power: the result of the number multiplied by itself twice:
$n^3 = n \times n \times n$.
Input Data:
Input a number: 8
*Expected Output*

```
Number is a cube.
```

**93 .** Write  a program to check a number is a cyclic or not.
A cyclic number is an integer in which cyclic permutations of the digits are successive multiples of the number. The most widely known are 142857:
142857 × 1 = 142857
142857 × 2 = 285714
142857 × 3 = 428571
142857 × 4 = 571428
142857 × 5 = 714285
142857 × 6 = 857142
Input Data:
Input a number: 142857
*Expected Output*

```
It is a cyclic number.
```

**94 W**rite a program to display first 10 Fermat numbers.
In mathematics, a Fermat number is a positive integer of the form

$$F_n = (F_{n-1} - 1)^2 + 1$$

where n is a nonnegative integer.
The first few Fermat numbers are:
3, 5, 17, 257, 65537, 4294967297, 18446744073709551617, …
*Expected Output*

```
3.0
5.0
17.0
257.0
65537.0
4.294967297E9
1.8446744073709552E19
3.4028236692093846E38
1.157920892373162E77
1.3407807929942597E154
Infinity
```

**95** Writeprogram to find any number between 1 and n that can be expressed as the sum of two cubes in two (or more) different ways.
//http://introcs.cs.princeton.edu/java/13flow/Ramanujan.java.html

Here are some examples of Ramanujan numbers :
1729 = 1^3 + 12^3 = 9^3 + 10^3
* 10000
1729 = 1^3 + 12^3 = 9^3 + 10^3
4104 = 2^3 + 16^3 = 9^3 + 15^3
* 100000
1729 = 1^3 + 12^3 = 9^3 + 10^3
4104 = 2^3 + 16^3 = 9^3 + 15^3
13832 = 2^3 + 24^3 = 18^3 + 20^3
39312 = 2^3 + 34^3 = 15^3 + 33^3
46683 = 3^3 + 36^3 = 27^3 + 30^3
32832 = 4^3 + 32^3 = 18^3 + 30^3
40033 = 9^3 + 34^3 = 16^3 + 33^3
20683 = 10^3 + 27^3 = 19^3 + 24^3
65728 = 12^3 + 40^3 = 31^3 + 33^3
64232 = 17^3 + 39^3 = 26^3 + 36^3

*Expected Output*

```
1729 = 1^3 + 12^3 = 9^3 + 10^3
4104 = 2^3 + 16^3 = 9^3 + 15^3
13832 = 2^3 + 24^3 = 18^3 + 20^3
39312 = 2^3 + 34^3 = 15^3 + 33^3
46683 = 3^3 + 36^3 = 27^3 + 30^3
32832 = 4^3 + 32^3 = 18^3 + 30^3
40033 = 9^3 + 34^3 = 16^3 + 33^3
20683 = 10^3 + 27^3 = 19^3 + 24^3
65728 = 12^3 + 40^3 = 31^3 + 33^3
64232 = 17^3 + 39^3 = 26^3 + 36^3
```

**96** Write a program to check if a number is Mersenne number or not.
In mathematics, a Mersenne number is a number that can be written in the form M(n) = $2^n - 1$ for some integer n.
The first four Mersenne primes are 3, 7, 31, and 127

*Expected Output*

```
Input a number: 127
127 is a Mersenne number.
```

**97** Write a program to find all the narcissistic numbers between 1 and 1000.
In number theory, a narcissistic number is a number that is the sum of its own digits each raised to the power of the number of digits.
For example:
$153 = 1^3 + 5^3 + 3^3$

*Expected Output*

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

**98** Write a program to print the first 15 numbers of the Pell series.
In mathematics, the Pell numbers are an infinite sequence of integers. The sequence of Pell numbers starts with 0 and 1, and then each Pell number is the sum of twice the previous Pell number and the Pell number before that.:
thus, 70 is the companion to 29, and $70 = 2 \times 29 + 12 = 58 + 12$.
The first few terms of the sequence are :
0, 1, 2, 5, 12, 29, 70, 169, 408, 985, 2378, 5741, 13860,…

*Expected Output*

```
First 20 Pell numbers:
1 2 5 12 29 70 169 408 985 2378 5741 13860 33461 80782 195025 470832 113
6689 2744210 6625109 15994428
```

**99.** Write a Program to check whether a number is a Keith Number or not.
In recreational mathematics, a Keith number or repfigit number (short for repetitive Fibonacci-like digit) is a number in the following integer sequence:
14, 19, 28, 47, 61, 75, 197, 742, 1104, 1537, 2208, 2580, 3684, 4788, 7385, 7647, 7909, 31331, 34285, 34348, 55604, 62662, 86935, 93993, 120284, 129106, 147640, 156146, 174680, 183186, 298320, 355419, 694280, 925993,

*Expected Output*

```
Input a number: 75
Keith Number
```

**100 W**rite a  program to create the first twenty Hamming numbers.
In computer science, regular numbers are often called Hamming numbers, Hamming Numbers are numbers whose only prime factors are 2, 3 and 5.
The first few hamming numbers are :
1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, 18, 20, 24, 25, 27, 30, 32

*Expected Output*

```
First Twenty Hamming numbers: 1 2 3 4 5 6 8 9 10 12 15 16 18 20 24 25 27
 30 32 36
```

1 WAP to swap two number Without using temp variable.

2 WAP to calculate power of a number using recurssion.

3 WAP to sum of digits of a number using recurssion.

4 WAP to convert decimal number to binary using recurssion .

5 A character is a vowel on consonant

6 A character is an alphabet or not

- ASCII value of a character
- Uppercase lowercase or special character
- Area of a circle
- Friendly pair or not
- Replace all zeros with 1 in a given integer
- Binary to decimal conversion
- Decimal to binary

- Decimal to octal
- Octal to decimal
- Binary to octal
- Octal to binary
- Maxm number of handshakes
- Quadrants in which coordinate lie
- Convert digit / number to words
- No of days in a given month of a given year
- Permutation in which n people can occupy r seats in theatre
- Nu of integers which has exactly 9 divisors
- Root of a quadratic equation
- Count possible decoding of a given digit sequence

- .

- . **Mathematical Algorithms:**

1. Write an Efficient Method to Check if a Number is Multiple of 3

2. Efficient way to multiply with 7
3. Write a C program to print all permutations of a given string
4. Lucky Numbers
5. Write a program to add two numbers in base 14
6. Babylonian method for square root
7. Multiply two integers without using multiplication, division and bitwise operators, and no loops
8. Print all combinations of points that can compose a given number
9. Write you own Power without using multiplication(*) and division(/) operators
10. Program for Fibonacci numbers
11. Average of a stream of numbers
12. Count numbers that don't contain 3
13. MagicSquare
14. Sieve of Eratosthenes
15. Number which has the maximum number of distinct prime factors in the range M to N
16. Find day of the week for a given date
17. DFA based division
18. Generate integer from 1 to 7 with equal probability
19. Given a number, find the next smallest palindrome
20. Make a fair coin from a biased coin
21. Check divisibility by 7
22. Find the largest multiple of 3
23. Lexicographic rank of a string
24. Print all permutations in sorted (lexicographic) order
25. Shuffle a given array
26. Space and time efficient Binomial Coefficient
27. Reservoir Sampling
28. Pascal's Triangle
29. Select a random number from stream, with O(1) space
30. Find the largest multiple of 2, 3 and 5
31. Efficient program to calculate e^x
32. Measure one litre using two vessels and infinite water supply
33. Efficient program to print all prime factors of a given number
34. Print all possible combinations of r elements in a given array of size n
35. Random number generator in arbitrary probability distribution fashion
36. How to check if a given number is Fibonacci number?
37. Russian Peasant Multiplication
38. Count all possible groups of size 2 or 3 that have sum as multiple of 3
39. Tower of Hanoi
40. Horner's Method for Polynomial Evaluation
41. Count trailing zeroes in factorial of a number
42. Program for nth Catalan Number
43. Generate one of 3 numbers according to given probabilities
44. Find Excel column name from a given column number
45. Find next greater number with same set of digits
46. Count Possible Decodings of a given Digit Sequence
47. Calculate the angle between hour hand and minute hand
48. Count number of binary strings without consecutive 1?s

49. Find the smallest number whose digits multiply to a given number n
50. Draw a circle without floating point arithmetic
51. How to check if an instance of 8 puzzle is solvable?
52. Birthday Paradox
53. Multiply two polynomials
54. Count Distinct Non-Negative Integer Pairs (x, y) that Satisfy the Inequality x*x + y*y < n
55. Count ways to reach the n'th stair
56. Replace all '0' with '5' in an input Integer
57. Program to add two polynomials
58. Print first k digits of 1/n where n is a positive integer
59. Given a number as a string, find the number of contiguous subsequences which recursively add up to 9
60. Program for Bisection Method
61. Program for Method Of False Position
62. Program for Newton Raphson Method

---

1. Find the element that appears once
2. Detect opposite sign
3. Set bits in all numbers from 1 to n

4 Swap bits

5 add two numbers

6. Smallest of three

7. A Boolean Array Puzzle

8. Set bits in an (big) array

9. Next higher number with same number of set bits

10. Optimization Technique (Modulus)

11. Add 1 to a number

12. Multiply with 3.5

13. Turn off the rightmost set bit

14. Check for Power of 4

15. Absolute value (abs) without branching

16. Modulus division by a power-of-2-number

17. Minimum or Maximum of two integers

18. Rotate bits

19. Find the two non-repeating elements in an array

20. Number Occurring Odd Number of Times

21. Check for Integer Overflow

22. Little and Big Endian

23. Reverse Bits of a Number

24. Count set bits in an integer

25. Number of bits to be flipped to convert A to B

26. Next Power of 2

27. Check if a Number is Multiple of 3

28. Find parity

29. Multiply with 7

30. Find whether a no is power of two

31. Position of rightmost set bit

32. Binary representation of a given number

33. Swap all odd and even bits

34. Find position of the only set bit

35. Karatsuba algorithm for fast multiplication

36. How to swap two numbers without using a temporary variable?

37. Check if a number is multiple of 9 using bitwise operators

38. Swap two nibbles in a byte

39. How to turn off a particular bit in a number?

40. Check if binary representation of a number is palindrome

---