Lab 3

1. The proper divisors of an integer n are defined as the positive divisors of n other than n itself, e.g. the proper divisors of 10 are 1, 2 and 5. A number is called a Perfect Number if the summation of all of its proper divisors is equal to the number itself. For example, 6 and 28 are perfect numbers because

1 + 2 + 3 = 6 and,

1 + 2 + 4 + 7 + 14 = 28.

It is found that there exist five such numbers 1, 6, 28, 496, 8128. Verify with your program output.

A number is called an Abundant Number if the summation of all of its proper divisors is greater than the number. For example, 12 is an abundant number because the sum of its divisors,

1 + 2 + 3 + 4 + 6 = 16 > 12

A number is called a Deficient Number if the summation of all of its proper divisors is less than the number. For example, 9 is a deficient number because the sum of its divisors,

1+3=4<9

Write a program that takes a positive integer n as input and prints whether it is “Perfect”, “Abundant” or “Deficient”.

Extend the program to find all three types of numbers in the range 1-9000.

Input:

Enter a number: 9

Output:

The number is Deficient Number

2. An Armstrong's(narcissistic) number (in base 10) of 3 digits is an integer such that the sum of the cubes of its digits is equal to the number itself. For example, 371 is an Armstrong number since $3^3 + 7^3 + 1^3 = 27 + 343 + 1=371$. 153 is an Armstrong number since $= 1^3 + 5^3 + 3^3= 1 + 125 + 27 = 153$.

Write a program to list all Armstrong numbers between 100 and 999. There exists 153, 370, 371, 407.

3. Given a Positive integer number n, write a complete program to find whether this number is strong number. A strong number is one whose sum of the factorials of each digit is equal to the number itself. Example: 145 is a strong number as its sum of factorial of each digits = 1!+4!+5!=1+24+120 = 145 which is the number itself], where as 154 or 165 are not a strong numbers.

Write a complete program to find all the strong numbers ranges between 1-30,000.

Extend the program to find first 3 strong numbers.

4. A happy number is defined by the following process: Starting with any positive integer, replace the number by the sum of the squares of its digits in base-ten, and

repeat the process until the number either equals 1 (where it will stay), or it loops endlessly in a cycle that does not include 1. Those numbers for which this process ends in 1 are happy numbers, while those that do not end in 1 are unhappy numbers. If a number is happy, then all members of its sequence are happy; if a number is unhappy, all members of the sequence are unhappy.
For example, 19 is happy, as the associated sequence is:
$1^2 + 9^2 = 82 \Rightarrow 8^2 + 2^2 = 68 \Rightarrow 6^2 + 8^2 = 100 \Rightarrow 1^2 + 0^2 + 0^2 = 1$
Write a program to print happy numbers in the range 1-1000
Your program should give following output.
1, 7, 10, 13, 19, 23, 28, 31, 32, 44, 49, 68, 70, 79, 82, 86, 91, 94, 97, 100, 103, 109, 129, 130, 133, 139, 167, 176, 188, 190, 192, 193, 203, 208, 219, 226, 230, 236, 239, 262, 263, 280, 291, 293, 301, 302, 310, 313, 319, 320, 326, 329, 331, 338, 356, 362, 365, 367, 368, 376, 379, 383, 386, 391, 392, 397, 404, 409, 440, 446, 464, 469, 478, 487, 490, 496, 536, 556, 563, 565, 566, 608, 617, 622, 623, 632, 635, 637, 638, 644, 649, 653, 655, 656, 665, 671, 673, 680, 683, 694, 700, 709, 716, 736, 739, 748, 761, 763, 784, 790, 793, 802, 806, 818, 820, 833, 836, 847, 860, 863, 874, 881, 888, 899, 901, 904, 907, 910, 912, 913, 921, 923, 931, 932, 937, 940, 946, 964, 970, 973, 989, 998, 1000
The happiness of a number is unaffected by rearranging the digits, and by inserting or removing any number of zeros anywhere in the number. The distinct combinations of digits that form happy numbers below 1,000 follow (the rest are just rearrangements and/or insertions of zero digits):
1, 7, 13, 19, 23, 28, 44, 49, 68, 79, 129, 133, 139, 167, 188, 226, 236, 239, 338, 356, 367, 368, 379, 446, 469, 478, 556, 566, 888, 899.
Write another program to print happy numbers containing distinct combinations of digits in the range 1-1000

5. Write a program to print prime numbers in the range 100-200
Write another program to print first 25 prime numbers

6. Write a program to print Fibonacci numbers in the range 1-1000. Check whether your program gives following output : 0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987
Write another program to print first 25 numbers in Fibonacci sequence

7. An integer is called perfect square if it can be expressed as square of some integer. For example, 1, 4, 9, 16 are perfect squares, but 6, 30, 44 are not perfect square.
We say than an integer is a square-free integer if it is not divisible by any perfect square number, except 1. For example, 30 is a square-free number since none of its divisors (other than 1), {2, 3, 5, 6, 10, 15, 30} is a perfect square. But 12 is a non-square-free number since its divisor 4 is a perfect square. Similarly, 10 is square-free but 18 is not, as it is divisible by 9 is square of 3 .The first few square-free numbers are 1,2,3,5,6,7,10,11,13.
Write a program to determine whether an integer is square-free.
Write a complete program to find first 20 square-free integers.
Input
n=20
Output
1, 2, 3, 5, 6, 7, 10, 11, 13, 14, 15, 17, 19, 21, 22, 23, 26, 29, 30, 31

8. The digital root (also known as repeated digital sum) of a number is the (single digit) value obtained by an repeated process of summing all digits, on each iteration using the result from the previous iteration to compute a digit sum. The process continues until a single-digit number is reached. Digital root(in base 10) of a number is the single digit obtained by repeatedly summing all the digits of a number.
Example:
Digital root of 2357 = 8
because (2 + 3 + 5 + 7 = 17) and (1 + 7 = 8)
Digital root of 89149 = 4
because (8 + 9 + 1 + 4 + 9 = 31) and (3 + 1 = 4)
Digital root of 45769486 is 4. When we add up the digits, we first get 4 + 5 + 7 + 6 + 9 + 4 + 8 + 6 = 49. Since 49 is not a single-digit number, we repeat the process on 49 once again. We get 4 + 9 = 13. 13 is not a single-digit number either, so we are forced to repeat the process one more time to get 1+ 3 = 4. 4 is the digital root of 45769486.
Digital root of 123 is 1+2+3=6
Digital Root of 93 is 3 (as we have cast out nine and the number 3 is the digital root.)
Digital root of 33572, we calculate first 3 + 3 + 5 + 7 + 2 = 20. The digital root of 33572 is then 2 = 2 + 0.
Write a complete program to calculate digital root and also find the number of steps to obtain the digital root.
Input
n=99
Output
Digital root of 99 is 9
Number of steps = 2

Digital root of a number is the single digit obtained by repeatedly summing all the digits of a number.
Example:
Digital root of 2357 = 8
because (2 + 3 + 5 + 7 = 17) and (1 + 7 = 8)
Digital root of 89149 = 4
because (8 + 9 + 1 + 4 + 9 = 31) and (3 + 1 = 4)

9. Write a program to input a range from user and print all the magic numbers in that range.
A number is magical if repeated adding of its digit gives 1. Example 19 is magical as 1 + 9 = 10, 1 + 0 = 1 hence magical. So is 991 as 9 + 9 + 1 = 19, 1 + 9 = 10, 1 + 0 = 1. However 224 is not.

10. A number n is called left truncatable prime if n and all numbers obtained by successively removing its left most digits are prime.(Similarly right truncatable prime is defined)
Ex 313 is a left truncatable prime >> 313 is prime, 13 is prime, 3 is prime.
313 is also right truncatable >> 313 is prime, 31 is prime, 3 is prime
Write a program which takes a number n as input and then tells whether it left truncatable, right truncatable or both.

11. Write Python programs to print the following patterns upto n lines:

a)
```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

b)
```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

c)
```
    *
   **
  ***
 ****
```

d)
```
****
***
**
*
```

e)
```
1
2 3
4 5 6
7 8 9 10
```

f)
```
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
```

g) 1
0 1
1 0 1
0 1 0 1
1 0 1 0 1

h) 1
2 6
3 7 10
4 8 11 13
5 9 12 14 15

i) 1
1 2 1
1 2 3 2 1
1 2 3 4 3 2 1

j) 1
2 3
3 4 5
4 5 6 7
5 6 7 8 9

k) 1
1 2
3 5 8
13 21 34 55
89 144 233 377 610

l) 1
2 6
3 7 10
4 8 11 13
5 9 12 14 15

m) *
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*

12. Write Python programs to print the following series upto n lines.
1) (1*1) + (2*2) + (3*3) + (4*4) + (5*5) + ... + (n*n)

2) (1) + (1+2) + (1+2+3) + (1+2+3+4) + ... + (1+2+3+4+...+n)

3) 1! + 2! + 3! + 4! + 5! + ... + n!

4) (1^1) + (2^2) + (3^3) + (4^4) + (5^5) + ... + (n^n)

5) (1!/1) + (2!/2) + (3!/3) + (4!/4) + (5!/5) + ... + (n!/n)

6) $x - x^3/3! + x^5/5! - x^7/7! + ...$ [input value of x(in Radian, convert it to degree) only, instead of n]

13. S. Ramanujan was an Indian mathematician who became famous for his intuition for numbers. When the English mathematician G. H. Hardy came to visit him in the hospital one day, Hardy remarked that the number of his taxi was 1729, a rather dull number. To which Ramanujan replied, "No, Hardy! No, Hardy! It is a very interesting number. It is the smallest number expressible as the sum of two cubes in two different ways." Verify this claim by writing a program in Python that takes an integer n(say, in the range 1-1000) from user and prints all integers less than or equal to n that can be expressed as the sum of two cubes in two (or more) different ways - find distinct positive integers a, b, c, and d such that $a^3 + b^3 = c^3 + d^3$. Use four nested for loops.
$1729 = 1^3 + 12^3 = 9^3 + 10^3$
$4104 = 2^3 + 16^3 = 9^3 + 15^3$