In [1]:

```
# Q1

def multiplication(k):
    '''
    This function compute the table of 'k'
    '''
    for i in range(1,11):
        print('{} x {} = {}'.format(k, i, k*i))

num = int(input('Enter the number: '))
multiplication(num)
```

```
Enter the number: 3
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
```

In [2]:

```
# Q2

def twin_prime(k):
    '''
    This function find the twin prime.
    If two consecutive odd numbers are prime then they are known as twin primes
    '''
    listprime = [2] # Create the list of prime numbers

    # Find the list of prime numbers less than 'k'
    for j in range(3, k):
        isDivisible = False
        for i in range(2, j):
            if (j%i == 0):
                isDivisible = True
                break

        if isDivisible == False:
            listprime.append(j)

    # Print the desire output
    for i in range(len(listprime)-1):
        if (listprime[i]%2 != 0) & (listprime[i+1]%2 != 0): # To check both are odd otherwise continue
            print([listprime[i], listprime[i+1]])

twin_prime(1000)
```

```
[3, 5]
[5, 7]
[7, 11]
[11, 13]
[13, 17]
[17, 19]
[19, 23]
[23, 29]
[29, 31]
[31, 37]
[37, 41]
[41, 43]
[43, 47]
[47, 53]
[53, 59]
```

```
[59, 61]
[61, 67]
[67, 71]
[71, 73]
[73, 79]
[79, 83]
[83, 89]
[89, 97]
[97, 101]
[101, 103]
[103, 107]
[107, 109]
[109, 113]
[113, 127]
[127, 131]
[131, 137]
[137, 139]
[139, 149]
[149, 151]
[151, 157]
[157, 163]
[163, 167]
[167, 173]
[173, 179]
[179, 181]
[181, 191]
[191, 193]
[193, 197]
[197, 199]
[199, 211]
[211, 223]
[223, 227]
[227, 229]
[229, 233]
[233, 239]
[239, 241]
[241, 251]
[251, 257]
[257, 263]
[263, 269]
[269, 271]
[271, 277]
[277, 281]
[281, 283]
[283, 293]
[293, 307]
[307, 311]
[311, 313]
[313, 317]
[317, 331]
[331, 337]
[337, 347]
[347, 349]
[349, 353]
[353, 359]
[359, 367]
[367, 373]
[373, 379]
[379, 383]
[383, 389]
[389, 397]
[397, 401]
[401, 409]
[409, 419]
[419, 421]
[421, 431]
[431, 433]
[433, 439]
[439, 443]
[443, 449]
[449, 457]
[457, 461]
[461, 463]
[463, 467]
[467, 479]
[479, 487]
[487, 491]
```

```
[491, 499]
[499, 503]
[503, 509]
[509, 521]
[521, 523]
[523, 541]
[541, 547]
[547, 557]
[557, 563]
[563, 569]
[569, 571]
[571, 577]
[577, 587]
[587, 593]
[593, 599]
[599, 601]
[601, 607]
[607, 613]
[613, 617]
[617, 619]
[619, 631]
[631, 641]
[641, 643]
[643, 647]
[647, 653]
[653, 659]
[659, 661]
[661, 673]
[673, 677]
[677, 683]
[683, 691]
[691, 701]
[701, 709]
[709, 719]
[719, 727]
[727, 733]
[733, 739]
[739, 743]
[743, 751]
[751, 757]
[757, 761]
[761, 769]
[769, 773]
[773, 787]
[787, 797]
[797, 809]
[809, 811]
[811, 821]
[821, 823]
[823, 827]
[827, 829]
[829, 839]
[839, 853]
[853, 857]
[857, 859]
[859, 863]
[863, 877]
[877, 881]
[881, 883]
[883, 887]
[887, 907]
[907, 911]
[911, 919]
[919, 929]
[929, 937]
[937, 941]
[941, 947]
[947, 953]
[953, 967]
[967, 971]
[971, 977]
[977, 983]
[983, 991]
[991, 997]
```

In [2]:

In [3]:

```
# Q3

def prime_factor(k):
    '''
    To find the prime factor of 'k'
    '''
    listfactor = []
    listprime = [2]

    # find the list of the prime numbers upto 'k'
    for j in range(3, k+1):
        isDivisible = False
        for i in range(2, j):
            if (j%i == 0):
                isDivisible = True
                break

        if isDivisible == False:
            listprime.append(j)

    l = 0
    # check every list of prime number is divisible by 'k'
    while (k >= 0) & (l < len(listprime)):
        if (k%listprime[l] == 0):
            listfactor.append(listprime[l])
            k = k / listprime[l]
        else:
            l += 1

    return listfactor

num = int(input('Enter the number: '))
print('Prime factor of this {} are {}'.format(num, prime_factor(num)))
```

```
Enter the number: 56
Prime factor of this 56 are [2, 2, 2, 7]
```

In [4]:

```
# Q4

def factorial(k):
    '''
    To find the factorial of 'k'
    '''
    f = 1
    if f < 0:
        f = 1
    else:
        for i in range(2,k+1):
            f = f*i

    return f

def permutation_(n, r):
    '''
    to find permunation with given (n,r)
    '''
    return factorial(n)/factorial(n-r)

def combination_(n,r):
    '''
    to find combination with given (n,r)
    '''
    return permutation_(n,r)/factorial(r)

num = int(input('Enter number n: '))
r = int(input('Enter number r: '))
print('Permutation of {} and {} is {}'.format(num,r,permutation_(num, r)))
print('Combination of {} and {} is {}'.format(num,r,combination_(num, r)))
```

```
Enter number n: 5
Enter number r: 2
Permutation of 5 and 2 is 20.0
Combination of 5 and 2 is 10.0
```

In [5]:

```python
# Q5

def decimal_to_binary(k):
    '''
    To convert decimal values to binary
    '''
    list_ = []

    # store the remainder value by dividing by 2
    while(k >= 1):
        list_.append(k%2)
        k = int(k / 2)

    # print reverse of the list we have got
    return reversed(list_)

num = int(input('Enter the number: '))
print('Binary representation of decimal value {} is {}'.format(num,list(decimal_to_binary(num))))
```

```
Enter the number: 121
Binary representation of decimal value 121 is [1, 1, 1, 1, 0, 0, 1]
```

In [6]:

```python
# Q6

def cubesum(k):
    '''
    To find the sum of cube of individual of that number
    '''
    _sum = 0
    while k > 0:
        # taking last unit digit
        r = k%10
        _sum += r**3
        # taking number except last digit
        k = int(k/10)

    return _sum

def PrintArmstrong(k):
    '''
    To print armstrong number
    '''
    _listarm = []

    # Finding the value that are armstrong number
    for i in range(k):
        _sum = cubesum(i)
        if _sum == i:
            _listarm.append(i)

    return _listarm

def isArmstrong(k):
    '''
    To check whether it is armstrong number or not
    '''
    # finding the sum of 'k'
    _sum = cubesum(k)

    # sum of cube of individual is equal to 'k' number, then it is armstrong number
    if _sum == k:
        print('{} is armstrong number'.format(k))
    else:
        # Otherwise not
```

```
        # Otherwise not
        print('{} is not armstrong number'.format(k))

num = int(input('Enter the number: '))
print('Cube Sum of {} is {}'. format(num,cubesum(num)))
print('Print Armstrong Number less than {} is {}'.format(num,PrintArmstrong(num)))
isArmstrong(num)
```

```
Enter the number: 371
Cube Sum of 371 is 371
Print Armstrong Number less than 371 is [0, 1, 153, 370]
371 is armstrong number
```

In [7]:

```
# Q7

def prodDigits(k):
    '''
    To calculate product of each digits
    '''
    _product = 1
    while k >= 1:
        # taking last unit digit
        r = k%10
        _product *= r
        # Taking number except last digit
        k = int(k/10)

    return _product

num = int(input('Enter the number: '))
print('Product of each digit in {} is {}'.format(num, prodDigits(num)))
```

```
Enter the number: 25164
Product of each digit in 25164 is 240
```

In [8]:

```
# Q8

def MDR(k):
    '''
    To find the multiplicative digital root of 'k'
    '''
    while(k > 10):
        k = prodDigits(k)

    return k

def MPersistence(k):
    '''
    To find the multiplicative persistence of 'k'
    '''
    count = 0
    while(k > 10):
        k = prodDigits(k)
        count += 1

    return count

num = int(input('Enter the number: '))
print('MDR {} and MPersistence {}'.format(MDR(num), MPersistence(num)))
```

```
Enter the number: 246
MDR 6 and MPersistence 3
```

In [9]:

```
# Q9
```

```
def sumPdivisors(k):
    '''
    To finds the sum of proper divisors of a number 'k'
    '''
    listdivisible = []
    _sum = 0
    for i in range(1,k):
        if k%i == 0:
            listdivisible.append(i)
            _sum += i

    return listdivisible, _sum

num = int(input('Enter the number: '))
x,y = sumPdivisors(num)
print('Proper divisor of {} are {} and their sum of proper divisor is {}'.format(num,x,y))
```

```
Enter the number: 56
Proper divisor of 56 are [1, 2, 4, 7, 8, 14, 28] and their sum of proper divisor is 64
```

In [10]:

```
# Q10

def find_perfectnum_range(x1,x2):
    '''
    To find the perfect number between x1 and x2
    '''
    list_perfect_num = [] # create empty list to store list of perfect number

    for i in range(x1,x2):
        # find the divisor of the 'i' number
        sumfactor, _sum = sumPdivisors(i)
        if _sum == i:
            list_perfect_num.append(i)

    return list_perfect_num

num1 = int(input('Enter number to find perfect number from: '))
num2 = int(input('Enter number to find perfect number to: '))
print(find_perfectnum_range(num1, num2))
```

```
Enter number to find perfect number from: 1
Enter number to find perfect number to: 2000
[6, 28, 496]
```

In [11]:

```
# Q11

def amicable_number_range(x1, x2):
    '''
    To find the pair of amicable number
    '''
    list_amicable_pair = []

    for i in range(x1, x2):
        # find the sum of proper divisor for 'i'
        _, _sum = sumPdivisors(i)
        # for the _sum above, again calculated to find the sum of proper divisor of _sum
        _, _sum1 = sumPdivisors(_sum)
        # If 'i' and '_sum1' same and if these pair not in the list rhen add in the list otherwise cont
inue
        if (i == _sum1) and ([i, _sum] not in list_amicable_pair) and ([_sum, i] not in list_amicable_p
air):
            list_amicable_pair.append([i, _sum])

    return list_amicable_pair

num1 = int(input('Enter number to find amicable number from: '))
num2 = int(input('Enter number to find amicable number to: '))
```

```
num2 = int(input("Enter number to find amicable number to: "))
print(amicable_number_range(num1, num2))
```

```
Enter number to find amicable number from: 1
Enter number to find amicable number to: 2000
[[6, 6], [28, 28], [220, 284], [496, 496], [1184, 1210]]
```

In [12]:

```
# Q12

num = range(1,20)
list(filter(lambda x: x%2 != 0, num))
```

Out[12]:

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

In [13]:

```
# Q13

num = range(1,20)

list(map(lambda x: x**3, num))
```

Out[13]:

```
[1,
 8,
 27,
 64,
 125,
 216,
 343,
 512,
 729,
 1000,
 1331,
 1728,
 2197,
 2744,
 3375,
 4096,
 4913,
 5832,
 6859]
```

In [14]:

```
# Q14

num = range(1,20)
list(map(lambda y: y**3 ,list(filter(lambda x: x%2 == 0, num))))
```

Out[14]:

```
[8, 64, 216, 512, 1000, 1728, 2744, 4096, 5832]
```

In [ ]:

```

```