In [5]:

```
#1. print multiplication table of a number

def table(num):
    '''
    This function prints the multiplication table for the given input
    '''
    for i in range(1, 13):
        print("{0} * {1} = {2}".format(num, i, num * i))

number = int(input("Enter a number to get its multiplication table : \n"))
table(number)
```

```
Enter a number to get its multiplication table :
5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
5 * 11 = 55
5 * 12 = 60
```

In [7]:

```
#2. Print twin prime numbers

def twinPrime(num):
    lst = []
    lst.append(1)
    for i in range(3, num, 2):
        count = 0
        for j in range(2, i):
            if 0 == i % j:
                count += 1

        if 0 == count:
            if 2 == i - lst[0]:
                print("{0} {1}".format(lst[0], i))
            lst[0] = i

number = int(input("Enter a range to print twin prime numbers : \n"))
twinPrime(number)
```

```
Enter a range to print twin prime numbers :
1000
1 3
3 5
5 7
11 13
17 19
29 31
41 43
59 61
71 73
101 103
107 109
137 139
149 151
179 181
191 193
197 199
227 229
239 241
269 271
```

```
281 283
311 313
347 349
419 421
431 433
461 463
521 523
569 571
599 601
617 619
641 643
659 661
809 811
821 823
827 829
857 859
881 883
```

In [6]:

```
#3. Program to print prime factors of a number

number = int(input("Enter a number to get its prime factors : \n"))
i = 2
while i < number + 1:
    if 0 == number % i:
        print(i)
        number /= i;
    else:
        i += 1
```

```
Enter a number to get its prime factors :
51
3
17
```

In [24]:

```
#4. Implement permutation and combination

def factorial(num):
    '''
    This function returns the factorial of a given input number
    '''
    return 1 if num <= 1 else num * factorial(num - 1)

def permutation(n, r):
    '''
    This function returns the permutation of a given number
    '''
    return factorial(n) / factorial(n - r)

def combination(n, r):
    '''
    This function returns the combination of a given number
    '''
    result = permutation(n, r) / factorial(r)
    print(result)
    return

n = int(input("Enter n for permutation and combination : \n"))
r = int(input("Enter r for permutation and combination : \n"))

result = permutation(n, r)
print(result)
combination(n, r)
```

```
Enter n for permutation and combination :
5
Enter r for permutation and combination :
5
120.0
```

```
1.0
```

In [41]:

```python
#5. Function to convert decimal to binary

def decToBin(num):
    '''
    This function converts the given decimal number into binary
    '''
    lst = []
    while num > 0:
        lst.append(str(num % 2))
        num = num >> 1
    return lst

number = int(input("Enter a number to convert into binary : \n"))
result = decToBin(number)
result.reverse()
print(''.join(result))
```

```
Enter a number to convert into binary :
99
1100011
```

In [14]:

```python
#6. Print Armstrong Numbers

def isArmstrong(baseNum, calculatedNum):
    '''
    This function checks if the parameter's value match each other to be an Armstrong number
    '''
    if baseNum == calculatedNum:
        return True
    else:
        return False

def cubesum(armStrongNum):
    '''
    This function calculates the armstrong number by cubing every individual number in the argument
    '''
    num = armStrongNum
    addition = 0
    while num > 0:
        addition += ((num % 10) ** 3)
        num = int(num / 10)
    return addition

def PrintArmstrong(rangeNum):
    '''
    This function prints all the Armstrong numbers belonging in the range provided
    '''
    for i in range(rangeNum):
        result = cubesum(i)
        armStrong = isArmstrong(i, result)
        if True == armStrong:
            print("The Number {0} is an Armstrong Number".format(i))
    return

number = int(input("Enter a range upto where the Armstrong Numbers should be printed : \n"))
PrintArmstrong(number)
```

```
Enter a range upto where the Armstrong Numbers should be printed :
1000
The Number 0 is an Armstrong Number
The Number 1 is an Armstrong Number
The Number 153 is an Armstrong Number
The Number 370 is an Armstrong Number
The Number 371 is an Armstrong Number
The Number 407 is an Armstrong Number
```

In [33]:

```
#7. Function that returns the product of every digit in the number

def prodDigits(num):
    product = 1
    while num > 0:
        product *= (num % 10)
        num = int(num / 10)
    return product

number = int(input("Enter a number : \n"))
product = prodDigits(number)
print(product)
```

```
Enter a number :
25
10
```

In [40]:

```
#8.
def MDR(num):
    return prodDigits(num)


def MPersistence(num, count):
    result = MDR(num)
    lst = list(str(result))
    count += 1
    if len(lst) > 1:
        MPersistence(result, count)
    else:
        print(result, count)

count = 0
MPersistence(341, count)
```

```
2 2
```

In [16]:

```
#9. Find the sum of divisor of a number

def sumPdivisors(num):
    count = 0
    for i in range(1, num):
        if 0 == num % i:
            count += i
    return count

number = int(input("Enter a number to get the divisor : \n"))
result = sumPdivisors(number)
print("Sum : ", result)
```

```
Enter a number to get the divisor :
500
Sum :  592
```

In [15]:

```
#10. Write a program to print all perfect numbers in a range

def perfectNumber(rangeNum):
    for i in range(1, rangeNum):
        total = 0
        for j in range(1, i):
            if 0 == i % j:
                total += j
        if i == total:
```

```
        if i == total:
            print("{0} is a perfect Number\n".format(i))

number = int(input("Enter the range in which to search perfect numbers : \n"))
perfectNumber(number)
```

```
Enter the range in which to search perfect numbers :
100
6 is a perfect Number

28 is a perfect Number
```

In [64]:

```
#11. Print pairs of amicable numbers
def divisorSum(num):
    count = 0
    for i in range(1, num):
        if 0 == num % i:
            count += i
    return count

def amicable(num):
    dct = dict()
    for i in range(1, num + 1):
        divisorCount = divisorSum(i)
        result = dct.get(divisorCount)
        if None == result:
            dct[i] = divisorCount
        else:
            if result == i:
                print("Amicable numbers : {0}, {1}".format(divisorCount, i))

number = int(input("Enter range to print the amicable number: \n"))
amicable(number)
```

```
Enter range to print the amicable number:
20000
Amicable numbers : 220, 284
Amicable numbers : 1184, 1210
Amicable numbers : 2620, 2924
Amicable numbers : 5020, 5564
Amicable numbers : 6232, 6368
Amicable numbers : 10744, 10856
Amicable numbers : 12285, 14595
Amicable numbers : 17296, 18416
```

In [57]:

```
#12. Write filter function using fiter\

def oddNumberFilter(num):
    if 1 == num % 2:
        return num

lst = list(range(-10, 10))
result = list(filter(oddNumberFilter, lst))
print(result)
```

```
[-9, -7, -5, -3, -1, 1, 3, 5, 7, 9]
```

In [18]:

```
#13. Write a function that returns cube of the numbers in a list

def cube(num):
    return num ** 3

lst = list(range(-10, 10))
result = list(map(cube, lst))
print(result)
print(result)
```

```
[-1000, -729, -512, -343, -216, -125, -64, -27, -8, -1, 0, 1, 8, 27, 64, 125, 216, 343, 512, 729]
```

In [32]:

```
#14. Write a program which can map() and filter() to make a list whose elements are cube of even number in a given list

import math

def cuberoot(num):
    result = math.ceil(abs(num) ** (1/3))
    if num < 0:
        result *= -1
    return result

def even(num):
    if 0 == num % 2:
        return num

lst = [-1000, -729, -512, -343, -216, -125, -64, -27, -8, -1, 0, 1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
cubeRootLst = list(map(cuberoot, lst))
evenLst = list(filter(even, cubeRootLst))
print(evenLst)
```

```
[-10, -8, -6, -4, -2, 2, 4, 6, 8, 10]
```

In [ ]: