In [147]:
```
#1.)Write a function that inputs a number and prints the multiplication table of that number


def multiplication(value,count):
    print(value, "x", count, "=", value*count)

    if count!=10:
        multiplication(value,count+1)#calling funtion using recursion

def checkIsIntValue():
    choice = ""
    while choice.isdigit()==False:
        choice = input("Enter a number:")


    if choice.isdigit()==False:
        print("Wrongly entered: ")
    else:
        return int(choice)

multiplication(checkIsIntValue(),1)
```

```
Enter a number:5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

In [148]:
```
#2.) Write a program to print twin primes less than 1000. If two consecutive odd numbers are both prime then they are known as twin primes


def checkPrime(maxNum):
    for i in range (2, maxNum):
        #Check given number is prime or not
        if maxNum % i == 0:
            return False
    return True

def twinPrime(maxNum):
    for i in range(3, maxNum):
        #Check for twin primes
        val = i + 2
        if (checkPrime(i) and checkPrime(val)):
            print(f" {i} and {val}")#show pattern learn

print('Twin Prime Number: \n')
twinPrime(checkIsIntValue())
```

```
Twin Prime Number:

Enter a number:1000
 3 and 5
 5 and 7
 11 and 13
 17 and 19
 29 and 31
 41 and 43
 59 and 61
 71 and 73
 101 and 103
 107 and 109
 137 and 139
 149 and 151
 179 and 181
 191 and 193
 197 and 199
 227 and 229
 239 and 241
 269 and 271
 281 and 283
 311 and 313
 347 and 349
 419 and 421
 431 and 433
 461 and 463
 521 and 523
 569 and 571
 599 and 601
 617 and 619
 641 and 643
 659 and 661
 809 and 811
 821 and 823
 827 and 829
 857 and 859
 881 and 883
```

In [149]:
```
#3.)write a program to find out the prime factors of a number. Example: prime factors of 56 - 2, 2, 2, 7

import math

lst = []

def primeFactors(value):
    while value % 2 == 0:
        lst.append(2)
        value = value/2


    for i in range(3, int(math.sqrt(value))+1, 2):
        while value%i == 0:
            lst.append(i)
            value = value/i


    if value > 2:
        value = int(value)
        lst.append(value)
    return lst

primeFactors(checkIsIntValue())
```

```
Enter a number:56
[2, 2, 2, 7]
```

Out[149]:

In [150]:
```
##.)write a program to implement these formulae of permutations and combinations.
def factorial(num):

    #factorial of a number
    if num == 1:
        return num
    return num * factorial(num-1)

def permutation(n, r):
    #permutation of a number
    return int(factorial(n) / factorial(n-r))

def combination(n, r):
    #combinations of a number
    return int(factorial(n) / (factorial(r) * factorial(n-r)))

print("Permutation:  ", permutation(15,4))
print("Combination:  ", combination(15,4))
```

```
Permutation:  32760
Combination:  1365
```

In [154]:
```
#5.)write a function that converts a decimal number to binary number
def decToBin(value):

    # Prints number of a given decimal number using recursion
    if value > 1:
        decToBin(value//2)
    print(value % 2, end="")

decToBin(checkIsIntValue())
```

```
Enter a number:11
1011
```

In [29]:
```
#6.)function of cubesum that accepts an integer and return the sum of the cubes
#of individual digits of that number. use this funtion printArmstrong() and is Armstrong
# to print Armstrong

def cubeOfSum(value):
    orderLength = len(str(num))
    cubeOfSum = 0
    while value > 0:
        digit = value % 10
        sumOfDigit += digit ** orderLength #Sum of individual digit and cube
        value //= 10


    # print("The Cube of a Given Number {0}  = {1}".format(value, cube))
    return int(sumOfDigit)



cubeOfNumber = checkIsIntValue()
valueOfSum = cubeOfSum(cubeOfNumber)

print("Sum of the digit",valueOfSum)
#Display the number is Armstrong Number

if cubeOfNumber == valueOfSum:
    print(cubeOfNumber,"Is An Armstrong Number")
else:
    print(cubeOfNumber,"Is Not an Armstrong Number")
```

```
Enter a number: 1634
Sum of the digit 1634
1634 Is an Armstrong Number
```

In [32]:
```
#7.)function prodDigit() that input a number and returns the product of digits of the number
def prodDigit(value):
    product = 1
    while value != 0:
        product = product * (value % 10)
        value = value // 10


    return product

print("Product of the digit of the number = ",prodDigit(checkIsIntValue()))
```

```
Enter a number: dsjfdhksj
Wrongly entered:
Enter a number: 45
Product of the digit of the number =  20
```

In [60]:
```
#8.)using the function prodDigits() of previous exercise write Funtions MDR() and MPersistence() that input a number and return its multiplication digital roo
```

```
def prodDigit(value):
    product = 1
    while value != 0:
        product = product * (value % 10)
        value = value // 10


    return product


def MDR(value):

    inputValue = str(value)
    mPersistence = 0
    while len(inputValue) > 1:
        inputValue = str(prodDigit(int(inputValue)))
        mPersistence += 1
    return int(inputValue), mPersistence

inputFrmUser = checkIsIntValue()
mdr, mper = MDR(inputFrmUser)
print("For {0} MDR is {1} and M Persistance is {2}".format(inputFrmUser, mdr, mper))
```

```
Enter a number: 11
For 11 MDR is 1 and M Persistance is 1
```

In [89]:
```
#9.)A funtion sumOfDivisors() that finds the sum of proper divisors of a number
#ex.)6 are 1,2,3,4,9,18
```

```
def sumOfDivisors(number):
    divisors = [1]
    for i in range(2, number):
        if number % i == 0:
            divisors.append(i)
    return sumOfDivisors()

def sumOfListDiv():
    sum=1
    for i in listDiv:
        sum+=i
    return sum


print("The sum of proper divisors of a number = ",sumOfDivisors(checkIsIntValue()))
```

```
Enter a number: 100
The sum of proper divisors of a number =  117
```

In [90]:
```
#10.)A number is called perfect if the sum of proper divisors of that number is equal to the number.
#for example 28 is perfect number, since 1+2+4+7+14=28.
#write a program to print all the perfect numbers in a given range
```

```
def perfectNums(upper):
    if upper< 1:
        return False
    perfect_sum = 0
    for i in range(1,upper):
        if sumOfDivisors(i) == i:
            perfect_sum += i


    return perfect_sum == upper

def checkIsIntValue():
    choice = ""
    while choice.isdigit()==False:
        choice = input("Enter a Range: ")


        if choice.isdigit()==False:
            print("Wrongly Range Number Entered: ")
        else:
            print("The perfect numbers in a given range:")
            return int(choice)

valueInt = checkIsIntValue()
for i in range(0, valueInt):
    if perfectNums(i):
        print(i, end=' ')
```

```
Enter a Range: 1000
The perfect numbers in a given range:
6 28
```

In [138]:
```
#11.) Two different numbers are called amicable numbers if the sum of the proper divisors of each is equal to the other number.
#for example 220 and 284 are amicable numbers.
def amicableNum(value1, value2):
    # amicable numbers in given range
    for i in range(value1, value2+1):
        for j in range(value1, value2+1):
            if i != j:
                if amicablePair(i, j):
                    print(i, j)

def amicablePair(first, second):
    #given pair is amicable or not
    return (sumOfDivisors(first) == second) and (sumOfDivisors(second) == first)



def checkIsIntValue():
    choice = ""
    while choice.isdigit()==False:
        choice = input("Enter a Range: ")


        if choice.isdigit()==False:
            print("Wrongly Range Number Entered: ")
        else:
            print("The Pairs of amicable numbers in a range :")
            return int(choice)

amicableNum(1, checkIsIntValue())
```

```
Enter a Range: 1000
The Pairs of amicable numbers in a range :
220 284
```

In [139]:
```
#12.) Write a program which can filter odd numbers in a list by using filter function
def filterOddNumber(value):
    #filter odd numbers from given list
    return list(filter(lambda i: (i%2 != 0), value))

listArray = []
choice='x'
while choice=='y' or choice=='Y':
    item=int(input('Enter the value in int to find the cube : '))
    listArray.append(item)
    choice = input('press y to continue else press any other key :')



filterOddNo(listArray)
```

```
Enter the value in int to find the cube : 5
Enter the value in int to find the cube : 2
Enter the value in int to find the cube : 3
press y to continue else press any other key :y
Enter the value in int to find the cube : 4
Enter the value in int to find the cube : 8
Enter the value in int to find the cube : 9
press y to continue else press any other key :2
```

Out[139]:
```
[5, 3]
```

In [134]:
```
#13.)write a program which can map() to make a list whose elements are cube of elements in a given list
def cubeNumber(list):
    return list(map(lambda x: x**3, lst))



listArray = []
choice='x'
while choice=='y' or choice=='Y':
    item=int(input('Enter the value in int to find the cube : '))
    listArray.append(item)
    choice = input('press y to continue else press any other key :')


cubeNumber(listArray)
```

```
Enter the value in int to find the cube : 2
press y to continue else press any other key :y

The list of cubes of given number
Enter the value in int to find the cube : 5
press y to continue else press any other key :y

The list of cubes of given number
Enter the value in int to find the cube : 4
press y to continue else press any other key :n

The list of cubes of given number
[8, 125, 64]
```

Out[134]:

In [135]:
```
#14.)Write a program which can map() and filter() to make a list whose elements are cube of even number in a given list
def cubeOfEven(lst):
    return cubeNumber(list(filter(lambda i: (num%2 == 0), lst)))



listArray = []
choice='x'
while choice=='y' or choice=='Y':
    item=int(input('Enter the value in int to find the cube : '))
    listArray.append(item)
    choice = input('press y to continue else press any other key :')
    if choice!='y' and choice!='Y':
        print("\nThe list of cubes of given number")
cubeOfEven(listArray)
```

```
Enter the value in int to find the cube : 4
press y to continue else press any other key :y

The list of cubes of given number
Enter the value in int to find the cube : 7
press y to continue else press any other key :y

The list of cubes of given number
Enter the value in int to find the cube : 8
press y to continue else press any other key :y

The list of cubes of given number
Enter the value in int to find the cube : 2
press y to continue else press any other key :y

The list of cubes of given number
Enter the value in int to find the cube : 7
press y to continue else press any other key :n

The list of cubes of given number
[64, 512, 8]
```

Out[135]:

In [136]:
```
print("End Of Optional Assignment")
```

```
End Of Optional Assignment
```

In [ ]: