In [7]:

```
#Write a function that inputs a number and prints the multiplication table of that number
def mul_table(n):
    for i in range(1,11):
        print("{}*{}={}".format(n,i,n*i))
    return
mul_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
```

In [17]:

```
#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 isprime(num):
    p=False
    for i in range(2,num):
        if num%i==0:
            p=True

    return not p

for k in range(3,1001):
    i=k
    j=k+2
    if i%2!=0 and j%2!=0:
        if isprime(i) and isprime(j):
            print((i,j))
```

```
(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 [18]:

```
#Write a program to find out the prime factors of a number. Example: prime factors of 56 -
#2, 2, 2, 7


def p_fac(n):
    l=[]

    i = 2
    while n > 1.0:
        if isprime(i):
            if "{0:.2f}".format(n/i) != "{0:.2f}".format(n // i):

                i = i + 1
            else:
                n=n/i
                l.append(i)
        else:
            i=i+1
    return l
print(p_fac(117))
```

```
[3, 3, 13]
```

In [10]:

```
#Write a program to implement these formulae of permutations and combinations.
#Number of permutations of n objects taken r at a time: p(n, r) = n! / (n-r)!. Number of
#combinations of n objects taken r at a time is: c(n, r) = n! / (r!*(n-r)!) = p(n,r) / r!

def fac(any):
    return 1 if any==1 else any*fac(any-1)
#print(fac(5))
n=int(input("enter n"))
r=int(input("enter r"))
p=fac(n)/fac(n-r)
print("p(n,r) =",p)
c=p/fac(r)
print("c(n,r) =",c)
```

```
enter n10
enter r9
p(n,r) = 3628800.0
c(n,r) = 10.0
```

In [11]:

```
#Write a function that converts a decimal number to binary number

def dectobin(n):
    l=[]

    while n>0:
        if n%2==0:
            r=n%2
            n=n//2
            l.append(str(r))
        elif n%2!=0:
            r=n%2
            n=n//2
            l.append(str(r))
    return "".join(l[::-1])
dectobin(33)
```

Out[11]:

```
'100001'
```

In [4]:

```
#Write a function cubesum() that accepts an integer and returns the sum of the cubes of
#individual digits of that number. Use this function to make functions PrintArmstrong() and
#isArmstrong() to print Armstrong numbers and to find whether is an Armstrong number.

def cubesum(num):
    sum=0
    for i in str(num):
        i=int(i)
        sum+=int(i*i*i)
    return sum

def isarmstrong(num):
    return True if cubesum(num)== num else False
def printarm(num):
    return str(num)+' is armstrong' if isarmstrong(num) else 'Not Armstrong'

#for i in range(1000):
#   if isarmstrong(i):
#       print(printarm(i))
print(printarm(153))
```

```
153 is armstrong
```

In [5]:

```
#Write a function prodDigits() that inputs a number and returns the product of digits of th
#number.

def proddigits(num):
    prod=1
    for i in str(num):
        i=int(i)
        prod*=i
    return prod
print(proddigits(345))
```

```
60
```

In [10]:

```
#If all digits of a number n are multiplied by each other repeating with the product, the c
#digit number obtained at last is called the multiplicative digital root of n. The number c
#times digits need to be multiplied to reach one digit is called the multiplicative
#persistance of n.
#Example: 86 -> 48 -> 32 -> 6 (MDR 6, MPersistence 3)
#341 -> 12->2 (MDR 2, MPersistence 2)
#Using the function prodDigits() of previous exercise write functions MDR() and
#MPersistence() that input a number and return its multiplicative digital root and
#multiplicative persistence respectively

def mdr(num):
    while (num>9):
        print(num,end=" ")
        num=proddigits(num)
    return num

def mper(num):
    c=0
    while (num>9):
        num=proddigits(num)
        c+=1
    return c

print(mper(86))
print(mdr(86))
```

```
3
86 48 32 6
```

In [9]:

```
#Write a function sumPdivisors() that finds the sum of proper divisors of a number. Proper
#divisors of a number are those numbers by which the number is divisible, except the
#number itself. For example proper divisors of 36 are 1, 2, 3, 4, 6, 9, 18

def sumpdiv(n):
    sum=0
    for i in range(1,n):
        if n%i==0:
            sum+=i
    return sum
print(sumpdiv(136))
```

```
134
```

In [11]:

```
#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 pernum(num):
    sum=0
    for i in range(1,num):
        if num%i==0:
            sum+=i
    if sum==num:
        return True
for i in range(1,10000):
    if pernum(i):
        print(i,end=' ')
```

```
6 28 496 8128
```

In [12]:

```
#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.
#Sum of proper divisors of 220 = 1+2+4+5+10+11+20+22+44+55+110 = 284
#Sum of proper divisors of 284 = 1+2+4+71+142 = 220
#Write a function to print pairs of amicable numbers in a range

def fun(n,n1):
    sum,sum1=0,0

    for i in range(1,n):
            if n%i==0:
                sum=sum+i
    for i in range(1,n1):
        if n1 % i == 0:
            sum1 = sum1 + i
    if n==sum1 and n1==sum:
        return True

for i in range(1,500):
    for j in range(1,500):
        if fun(i,j):
           print(i,j)
```

```
6 6
28 28
220 284
284 220
496 496
```

In [13]:

```
#Write a program which can filter odd numbers in a list by using filter function
l=list(filter(lambda x:x%2!=0, range(20)))
print(l)
```

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

In [14]:

```
#Write a program which can map() to make a list whose elements are cube of elements in
#a given list
l=list(map(lambda x:x*x*x, range(10)))
print(l)
```

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

In [15]:

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

l=list(map(lambda x:x*x*x, range(10)))
s=list(filter(lambda x:x%2==0, l))
print(s)
```

```
[0, 8, 64, 216, 512]
```

In [ ]: