Question-1: Write a function that inputs a number and prints the multiplication table of that number.

In [14]:
```
def table(a):
    """
    Multiplication Table of the input number
    """
    for i in range(1,11):
        c = i*a
        print(a,"*",i,"=",c)

m=input("write the number: ")
print("Multiplication table of",m)
table(int(m))
```

```
write the number: 21
Multiplication table of 21
21 * 1 = 21
21 * 2 = 42
21 * 3 = 63
21 * 4 = 84
21 * 5 = 105
21 * 6 = 126
21 * 7 = 147
21 * 8 = 168
21 * 9 = 189
21 * 10 = 210
```

Question-2: Write a program to print twin primes less than 1000. If two consecutive odd numbers both prime then they are known as twin primes

In [15]:
```
print("Twin primes below 1000 are given below:")
for num in range (3,1000):
        isDiv = False;
```

```
        for dig in range(2,num):
            if num%dig == 0:
                isDiv = True;

        if not isDiv:
            num_2 = num+2                #checking the consecutive odd
 number to be prime
            for dig_2 in range(2,num_2):
                if num_2%dig_2 == 0:
                    isDiv = True;

            if not isDiv:
                print(num, num_2)     #printing only when the consecuti
ve number is prime
```

```
Twin primes below 1000 are given below:
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
```

Question-3: Write a program to find out the prime factors of a number. Example: Prime factor of 56- 2, 2, 2, 7

In [16]:
```
pfactors = []
def prime_factor(n):
    for i in range(2,int(n+1)):
        if (n%i==0):
            pfactors.append(i)
            n = n/i
            break

    if n>1:                        #using recursive function
        prime_factor(n)
    else:
        print(pfactors)

m = int(input("Write the number: "))
print("prime factorization of", m)
prime_factor(m)
```

```
Write the number: 76
prime factorization of 76
[2, 2, 19]
```

Question-4: 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: c(n,r) = n! / (r!*(n-r)!) = p(n,r) / r!

In [17]:
```
def factorial(n):
    return 1 if n==1 else (n*factorial(n-1))

n = int(input("Write number of objects: "))
r = int(input("Number of times the object is taken? "))

per = factorial(n)/factorial(n-r)
comb = per/factorial(r)
print("*_*_*_*_*_*_*_*_*_*")
print("Number of permutations of {} objects taken {} at a time is: {}".
format(n, r, per))
print("Number of combinations of {} objects taken {} at a time is: {}".
format(n, r, comb))
print("*_*_*_*_*_*_*_*_*_*")
```

```
Write number of objects: 10
Number of times the object is taken? 4
*_*_*_*_*_*_*_*_*_*
Number of permutations of 10 objects taken 4 at a time is: 5040.0
Number of combinations of 10 objects taken 4 at a time is: 210.0
*_*_*_*_*_*_*_*_*_*
```

Question-5: Write a function that converts decimal number to a binary number.

In [18]:
```
n= int(input("Write the decimal number: "))

lst = []

while n>0:
```

```
    lst.append(n%2)
    n= int(n/2)

lst.reverse()
print("binary form:",end=" ")
for i in lst:
    print(i,end="")
```

```
Write the decimal number: 183
binary form: 10110111
```

Question-6: Write a function cubesum() that accepts an integer and returns the sum of the cubes of the individual digits of that number. Use this function to make functions PrintArmstrong() and isArmstrong() to print Armstrong numbers and to find out whether is an Armstrong number.

In [19]:
```
def cubesum(num):
    st = str(num)
    lst=[]
    for i in st:
        s = int(i)**3
        lst.append(s)
    return (sum(lst))

def PrintArmstrong(n):
    for x in range(n):
        if (cubesum(x)==x):
            print(x)

def isArmstrong(m):
    return True if (cubesum(m)==m) else False

n = int(input("Write a number to print armstrong number less than it: "
))
PrintArmstrong(n)
print('\n')
m = int(input("write a number to check if it is an Armstrong number: "
))
isArmstrong(m)
```

```
Write a number to print armstrong number less than it: 400

0
1
153
370
371


write a number to check if it is an Armstrong number: 371
```

Out[19]:
```
True
```

Question-7: Write a function prodDigits() that inputs a number and returns the product of digits of that number.

In [5]:
```
def prodDigits(num):
    product = 1;
    for i in num:
        i=int(i)
        product*=i
    return product

num = input("Type the number: ")
prodDigits(num)
```

```
Type the number: 56
```

Out[5]:
```
30
```

Question-8: If all digits of a number n are multiplied by each other repeating with the product, the one digit number obtained at last is called the multiplicative digit root of n. The number of times digit need to be multiplied to reach one digit is called the multiplicative persistance of n. Example: 86->48->32->6 (MDR 6, MPersistance 3) 341->12->2 (MDR 2, MPersistance 2) Using the

function prodDigits() of previous exercise write functions MDR() and MPersistance() that input a number and return its multiplicative digital root and multiplicative persistance respectively.

In [20]:
```
def prodDigits(num):
    product = 1;
    for i in num:
        i=int(i)
        product*=i
    return product

def MDR(mdr):
    s = prodDigits(mdr)
    s = str(s)
    return s if (len(s)==1) else MDR(s)

count=[]
def MPersistance(mdr):
    count.append(1)
    s = prodDigits(mdr)
    s = str(s)
    return sum(count) if (len(s)==1) else MPersistance(s)


mdr =(input("Type the number: "))
print("MDR of {} is {}".format(mdr, MDR(mdr)))
print("MPersistance of {} is {}".format(mdr, MPersistance(mdr)))
```

```
Type the number: 873
MDR of 873 is 6
MPersistance of 873 is 4
```

Question-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, 12,18

In [21]:
```
def sumPdivisors(num):
    for i in range (1,num):
```

```
        if (num%i==0):
            divisors.append(i)
    return sum(divisors)
divisors = []
n = int(input("Type the number: "))
print("Sum of proper divisors of {} is {}".format(n, sumPdivisors(n)))
```

```
Type the number: 22
Sum of proper divisors of 22 is 14
```

Question-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

In [22]:
```
def sumPdivisors(num):
    divisors = []                #divisor list put inside to avoid app
ending of old values.
    for i in range (1,num):
        if (num%i==0):
            divisors.append(i)
    return sum(divisors)

z = int(input("Write a number: "))
print("Perfect number below", z)

for x in range(z):
    if x==sumPdivisors(x):
        print(x)
```

```
Write a number: 32
Perfect number below 32
0
6
28
```

Question-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.

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

In [23]:
```
def sumPdivisors(num):
    divisors = []                #divisor list put inside to avoid app
ending of old values.
    for i in range (1,num):
        if (num%i==0):
            divisors.append(i)
    return sum(divisors)
z =  int(input("Write the range: "))
for x in range(z) :
    for y in range(z):
        if (x==sumPdivisors(y) and y==sumPdivisors(x) and x!=y):
            print(x,y)
```

```
Write the range: 500
220 284
284 220
```

Question-12: Write a program which can filter odd numbers in a list by using filter function

In [9]:
```
lst=[1,2,3,4,5,6,7,8,9,10]
odd_lst= list(filter(lambda x: x%2!=0, lst))
print("original list:",lst)
print("\nodd list:",odd_lst)
```

```
original list: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

odd list: [1, 3, 5, 7, 9]
```

Question-13: Write a program which can map() to make a list whose elements are cube of elements in a given list

In [12]:
```
lst= [1,2,3,4,5,6,7,8,9,10]
cube=list(map(lambda x: x**3, lst))
print("Original list:", lst)
print("\nCube List:", cube)
```

```
Original list: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Cube List: [1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

Question-14: Write a program which can map() and filter() to make a list whose elements are cube of even number in a given list

In [13]:
```
lst = [5,6,7,8,9,101,103,105,106,200]
even_lst= list(filter(lambda x: x%2==0, lst))
cube_even=list(map(lambda x: x**3, even_lst))
print("Original list:", lst)
print("\nCube of even:", cube_even)
```

```
Original list: [5, 6, 7, 8, 9, 101, 103, 105, 106, 200]

Cube of even: [216, 512, 1191016, 8000000]
```

End of Assignment op1

In [ ]:
```

```