1. Write a function that input a number and prints the multiplication table of that number

In [33]:
```
def printTable():
    """
        print multiplication table from 1 to n
    """
    try:
        num = int(input("Enter number"))
        for i in range(1,11):
            print("{} * {} =  {}".format(num,i,num*i))
    except(ValueError):
        print("Error! enter number")
printTable()
```

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

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

In [23]:
```
#print twin primes if to consecutive odd numbers are print they are
known as twin primes
#solution: First find the primeList and then check if number is odd
and two number's difference is 2;
primeList=[]
for i in range(2,1001):
    isPrime=True;
    for j in range(2,i):
        if i%j==0:
            isPrime=False
            break
    if isPrime:
        primeList.append(i)

pre=-1;
twinPrimeList=[]
for i in primeList:
   # print("{},{}".format(pre,i))
    if (i-pre)==2:
        print("{},{}".format(pre,i))
    pre=i
```

```
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
```

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

In [40]:
```
import math
def isPrime(num):
    """
    check if given number is prime or not.
    If we check the prime condition till square root of a number we
can get the result
    """
    sqrt = int(math.sqrt(num))+1
    #print("SQRT of {} is {}".format(num,sqrt))
    for i in range(2,sqrt+1):
        if num%i==0 and num!=2:
            return False
    return True;

num=0
try:
    num=int(input("Enter Number:"))
    lst=[]
    if isPrime(num):
        lst.append(num)
        print(lst)
    else:
        i=2
        sqrt=int(math.sqrt(num))+1
        print("sqrt:",sqrt)
        while num>=i:
            if num%i==0:
                num=num/i;
                if isPrime(i):
                    lst.append(i);
            else:
                i+=1
            #print("{},{}".format(i,num))
        print(lst);
except(ValueError):
    print("ERROR: Please Enter Number")
```

```
Enter Number:32
sqrt: 6
[2, 2, 2, 2, 2]
```

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

In [53]:
```
from functools import reduce
def computeP(n,r):
    """
    compute nPr : n!/(n-r)!
    """
    nfact=reduce(lambda x,y: (x*y),range(1,n+1))
    #dfact stands for difference of (n-r)!
    dfact=reduce(lambda x,y: x*y,range(1,(n-r)+1))
    #print(dfact)
    return nfact/dfact

def computeC(n,r):
    """
    compute nCr : n!/(r!*(n-r)!) ==> nPr/r!
    """
    rfact=reduce(lambda x,y:(x*y),range(1,r+1))
    npr=computeP(n,r);
    return npr/rfact;

c=computeC(5,3)
print("C(n,r)={}".format(c))
```

```
C(n,r)=10.0
```

5. Write a function that converts a decimal number to binary number

In [22]:
```
def decimalToBinary(n):
    lst=[]
    while n>0:
        r=int(n%2)
        lst.insert(0,r)
        n=int(n/2);
    lst.insert(0,n)
    return lst
d = int(4)
print(decimalToBinary(d))
```

```
[0, 1, 0, 0]
```

6. 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.

In [45]:
```
def cubesum(n):
    t=n
    _sum=0;
    while n>0:
        val=int(n%10);
        #print("v={},v**3={}".format(val,val**3))
        val=val**3;
        _sum+=val;
        n=int(n/10)
    #print("n={},_sum={}".format(t,_sum))
    return _sum
def isArmstranong(n):
    cs=cubesum(n)
    if cs==n:
        return True;
    else :
        return False;
def printArmstrong(n):
    if isArmstranong(n):
        return n
    else:
        return 0
lst = list(filter(printArmstrong,range(1,1001)))
print(lst)
```

```
[1, 153, 370, 371, 407]
```

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

```
In [24]: def prodDigits(n):
             s=1
             while n!=0:
                 r=int(n%10)
                 #print(r)
                 s*=r
                 n=int(n/10)
             return s;
         n = int(input("enter number:"))
         res = prodDigits(n)
         print("prod digit:",res)
```

```
enter number:-86
prod digit: 8
```

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 digital root of n. The number of 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

In [30]:
```
def MDR(n):
    while n>9:
        n=prodDigits(n)
    return n;
def MPersistence(n):
    count=0
    while n>9:
        count+=1;
        n=prodDigits(n)
    return count


n=int(input("please enter number:"))
isNeg=False;
if n<0:
    n=abs(n);
    isNeg=True
res = MDR(n)
sign=""
if isNeg:
    sign="-";

print("res: {}{}".format(sign,res))

mPersist=MPersistence(n)
print("mPersist:",mPersist)
```

```
please enter number:-86
res: -6
mPersist: 3
```

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

```
In [60]: import math
         def sumPdivisors(num):
             sm=1
             sqrt=int(math.sqrt(num));
             #print(sqrt)
             for i in range(2,sqrt+1):
                 if num%i==0:
                     sm+=i
                     sm+=int(num/i)

             return sm

         num=223#284
         sm = sumPdivisors(num)
         print(sm)

         1
```

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 [26]: def allPerfectNumber(num):
             return sumPdivisors(num)==num
         lst = list(filter(allPerfectNumber,range(1,10000)))
         print(lst)

         [1, 6, 28, 496, 8128]
```

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 [63]:
```
def printAmicableNumbers(limit):
    _set=set()
    lst=[]
    for i in range(200,300):
        n=sumPdivisors(i);
        if n in _set and n!=1:
            _set.remove(n)
            _set.discard(i)
           # print("{} in set {}".format(n,i));
            lst.append((n,i))
        else:
            _set.add(i)
            _set.add(n)
        #print("n={} and i={}".format(n,i))
    return lst
limit=301
lst = printAmicableNumbers(limit)
lst.sort()
print(lst)
#print(dir(set))
```

```
[(31, 221), (35, 289), (37, 299), (49, 287), (97, 275), (202, 230)
, (208, 268), (214, 266), (218, 232), (218, 286), (220, 284), (226
, 292), (250, 290), (274, 296)]
```

12. Write a program which can filter odd numbers in a list by using filter function

In [43]:
```
lst = list(filter((lambda x: x%2==1),range(1,1001)))
print(lst)
```

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35
, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67,
69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99, 10
1, 103, 105, 107, 109, 111, 113, 115, 117, 119, 121, 123, 125, 127
, 129, 131, 133, 135, 137, 139, 141, 143, 145, 147, 149, 151, 153,
155, 157, 159, 161, 163, 165, 167, 169, 171, 173, 175, 177, 179, 1
81, 183, 185, 187, 189, 191, 193, 195, 197, 199, 201, 203, 205, 20
7, 209, 211, 213, 215, 217, 219, 221, 223, 225, 227, 229, 231, 233
, 235, 237, 239, 241, 243, 245, 247, 249, 251, 253, 255, 257, 259,
261, 263, 265, 267, 269, 271, 273, 275, 277, 279, 281, 283, 285, 2
87, 289, 291, 293, 295, 297, 299, 301, 303, 305, 307, 309, 311, 31
3, 315, 317, 319, 321, 323, 325, 327, 329, 331, 333, 335, 337, 339
, 341, 343, 345, 347, 349, 351, 353, 355, 357, 359, 361, 363, 365,
367, 369, 371, 373, 375, 377, 379, 381, 383, 385, 387, 389, 391, 3
93, 395, 397, 399, 401, 403, 405, 407, 409, 411, 413, 415, 417, 41
9, 421, 423, 425, 427, 429, 431, 433, 435, 437, 439, 441, 443, 445
, 447, 449, 451, 453, 455, 457, 459, 461, 463, 465, 467, 469, 471,
473, 475, 477, 479, 481, 483, 485, 487, 489, 491, 493, 495, 497, 4
99, 501, 503, 505, 507, 509, 511, 513, 515, 517, 519, 521, 523, 52
5, 527, 529, 531, 533, 535, 537, 539, 541, 543, 545, 547, 549, 551
, 553, 555, 557, 559, 561, 563, 565, 567, 569, 571, 573, 575, 577,
579, 581, 583, 585, 587, 589, 591, 593, 595, 597, 599, 601, 603, 6
05, 607, 609, 611, 613, 615, 617, 619, 621, 623, 625, 627, 629, 63
1, 633, 635, 637, 639, 641, 643, 645, 647, 649, 651, 653, 655, 657
, 659, 661, 663, 665, 667, 669, 671, 673, 675, 677, 679, 681, 683,
685, 687, 689, 691, 693, 695, 697, 699, 701, 703, 705, 707, 709, 7
11, 713, 715, 717, 719, 721, 723, 725, 727, 729, 731, 733, 735, 73
7, 739, 741, 743, 745, 747, 749, 751, 753, 755, 757, 759, 761, 763
, 765, 767, 769, 771, 773, 775, 777, 779, 781, 783, 785, 787, 789,
791, 793, 795, 797, 799, 801, 803, 805, 807, 809, 811, 813, 815, 8
17, 819, 821, 823, 825, 827, 829, 831, 833, 835, 837, 839, 841, 84
3, 845, 847, 849, 851, 853, 855, 857, 859, 861, 863, 865, 867, 869
, 871, 873, 875, 877, 879, 881, 883, 885, 887, 889, 891, 893, 895,
897, 899, 901, 903, 905, 907, 909, 911, 913, 915, 917, 919, 921, 9
23, 925, 927, 929, 931, 933, 935, 937, 939, 941, 943, 945, 947, 94
9, 951, 953, 955, 957, 959, 961, 963, 965, 967, 969, 971, 973, 975
, 977, 979, 981, 983, 985, 987, 989, 991, 993, 995, 997, 999]
```

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

In [48]:
```
lst=list(map((lambda x:x**3),range(1,20)))
print(lst)
```

```
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000, 1331, 1728, 2197, 27
44, 3375, 4096, 4913, 5832, 6859]
```

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 [49]:
```
lst=list(filter((lambda x:x%2==0),range(1,20)))
print("Filter : ",lst)
lst=list(map((lambda x:x**3),lst))
print("map : ",lst)
```

```
Filter :  [2, 4, 6, 8, 10, 12, 14, 16, 18]
map :  [8, 64, 216, 512, 1000, 1728, 2744, 4096, 5832]
```