**Assignment 1**

## Q.1 Write a function that inputs a number and prints the multiplication table of that number

In [8]:

```
def mult(n):
    i=1
    while(i<11):
        print(i*n)
        i+=1
mult(int(input()))
```

```
10
10
20
30
40
50
60
70
80
90
100
```

## 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 [4]:

```
from math import sqrt
from math import gcd
n=1001
m=[]
l=[True]*(n)
l[0]=l[1]=False
for i in range(2,n):
    if(l[i]==True):
        for j in range(2,(n-1)//i+1):
            l[i*j]=False
for i in range(1,n):
    if(l[i]==True):
        m.append(i)
        #print(i)
print('All twin prime numbers under 1000')
for i in range(1,len(m),1):
    if(m[i]-m[i-1]==2):
        print(m[i-1],m[i])



#    for i in range(1,n,1):
#        for j in range(i+1,n-1,1):
#            if(l[i]==1 and l[j]==1 and gcd(l[i],l[j])==1):
#                print(i,"",j,end='  ')
#i make the above line as comment which it generate a very large output and when i am going to con
vert this into html to pdf
#it causes problem you can run it on your pc it work
```

```
All twin prime numbers under 1000
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
```

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

In [10]:

```
from math import sqrt
n=int(input())
l=[]
while(n>1):
    for i in range(2,n+1,1):
        if(n%i==0):
            l.append(i)

            n=n//l[-1]
            break
           # print(n)
            #print('sonu')
l.sort(reverse=False)
print(l)
```

```
56
[2, 2, 2, 7]
```

## Q.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 [11]:

```
from math import factorial
def permutation(n,r):
    return ( factorial(n)//(factorial(n-r)) )
def combination(n,r):
    return ( factorial(n)//(factorial(n-r)*factorial(r)))
n=int(input())
r=int(input())
print(permutation(n,r))
print(combination(n,r))
```

```
10
2
90
45
```

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

In [12]:

```
def decimal_to_binary(n):
    l=[]
    while(n>0):
        if(n%2==1):
            l.append(1)
        else:
            l.append(0)
        n//=2
    return (l[::-1])
print(decimal_to_binary(int(input())))
```

```
65
[1, 0, 0, 0, 0, 0, 1]
```

## Q.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 [15]:

```
def cubesum(n):
    sum=0
    while(n>0):
        p=n%10
        sum=sum+(p*p*p)
        n//=10
    return sum
n=int(input())
d=cubesum(n)
if(d==n):
    print('Yes this {} is a armstrong number'.format(n))
else:
    print('No this is {} not a armstrong number'.format(n))
```

```
153
Yes this 153 is a armstrong number
```

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

In [14]:

```
def prodDigit(n):
    p=1
    while(n>0):
        p=(p)*(n%10)
        n//=10
    return p
print(prodDigit(int(input())))
```

```
123456789
362880
```

## Q.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 [16]:

```
n=int(input())
while(True):
    m=n
    p=1
    while(m>0):
        p=p*(m%10)
        m//=10
    n=p
    if(n<10):
        break
print(n)
```

```
341
2
```

## Q.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 [17]:

```
def sumPdivisors(n):
    sum=0
    for i in range(1,int(sqrt(n)+1),1):
        if(n%i==0):
            sum=sum+i+n//i
        if(sqrt(n)==i):
            sum=sum-i
    return (sum-n)
from math import sqrt
print(sumPdivisors(int(input())))
```

```
36
55
```

## Q.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 [18]:

```
from math import sqrt
def perfect_no(n):
    sum=0
    for i in range(1,int(sqrt(n))+1,1):
        if(n%i==0):
```

```
            sum=sum+i+n//i
        if(sqrt(n)==i):
            sum-=i
    if((sum-n)==n):
        return True
    else:
        return False
n=int(input())
l=[]
for i in range(2,n+1,1):
    b=perfect_no(i)
    if(b==True):
        l.append(i)
print(l)
```

```
28
[6, 28]
```

## Q.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

In [19]:

```
from math import sqrt
n=int(input())
m=int(input())
l1=[]
l2=[]
for i in range(1,int(sqrt(n)+1),1):
        if(n%i==0):
            l1.append(i)
            l1.append(n//i)
for i in range(1,int(sqrt(m)+1),1):
    if(m%i==0):
        l2.append(i)
        l2.append(m//i)
l1.append(-n)
l2.append(-m)
if(sum(l1)==m and sum(l2)==n):
    print('amicable')
else:
    print('not amicable')
#print(l1)
#print(l2)
```

```
220
284
amicable
```

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

In [21]:

```
l=list( map(int,(input().split())) )
l1=list( filter( (lambda x:x%2==1) ,l) )
print(l1)
```

```
1 2 3 4 5 6 7 8 9
[1, 3, 5, 7, 9]
```

## Q.13 Write a program which can map() to make a list whose elements are cube

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

In [22]:

```
def cube(n):
    return(n*n*n)
l=list(map(int,input().split()))
l1=list(map(cube,l))
print(l1)
```

```
 1 2 3 4 5 6 7 8 9
[1, 8, 27, 64, 125, 216, 343, 512, 729]
```

## Q.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 [23]:

```
def cube(n):
    return(n*n*n)
l=list(map(int,input().split()))
l1=filter(lambda x:x%2==0,l)
l2=list(map(cube,l1))
print(l2)
```

```
1 2 3 4 5
[8, 64]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: