**1 : Arrays and Loops**

**AASTHA SOOD**

**16BCE1104**

**Q1 COMPUTING PRIME FACTORS**

Design algorithms for computing all the prime factors of an integer n. Note that every integer can be expressed as a product of prime numbers. Determine the time and space requirements of your algorithms. Implement your algorithms in Python.

**Algorithm:**

1. Define function **prime**(n):
2. Set=t **h**<-1
3. For **i** in range from 2 to **n**:
4. If **n** divisible by **i**:
5. Set **h**<-0
6. END
7. If **h** is 1:
8. Return TRUE
9. Input n
10. For i in range from 2 to n+1:
11. If (n is divisible by i) and (i is prime):
12. Print i

**Space and Complexity:**

1. Space required:

|  |  |
| --- | --- |
| VARIABLE | DATA TYPE |
| h | int(intger) |
| n | int(intger) |
| i | int(intger) |

b) Time complexity:

1. C1\* 1

2. C2\* 1

3. C3\*(N-2)

4. C4\* (N-3)

5. C5\*1

6. 0\*1

7. C6\*1

8. 0\*1

1. 0\*1

2. C7\*(N-1)

3. C8 \*(N-2)

4. C9 \* Σ 2n tn // tn=Number of time the if is satisfied

So Time Complexity,

T(n) = C7\*(N-1) + C8 \*(N-2) + (C 1+C 2+C3\*( N-2)+C4\* (N-3)+C5+C6)\*Σ 2n tn + C9\* Σ 2n tn

= C1\*(N-1) + C2 \*(N-2) + (C 3\*( N-2)+C4\* (N-3))\*Σ 2n tn + C4 \* Σ 2n tn

**Python code:**

def primeNumber(num):

h=1;

for i in range(2,num):

if num%i==0:

h=0

break

if h==1:

return True

num=int(input('Enter a number : '))

print('The prime factors of number',num,'are : ')

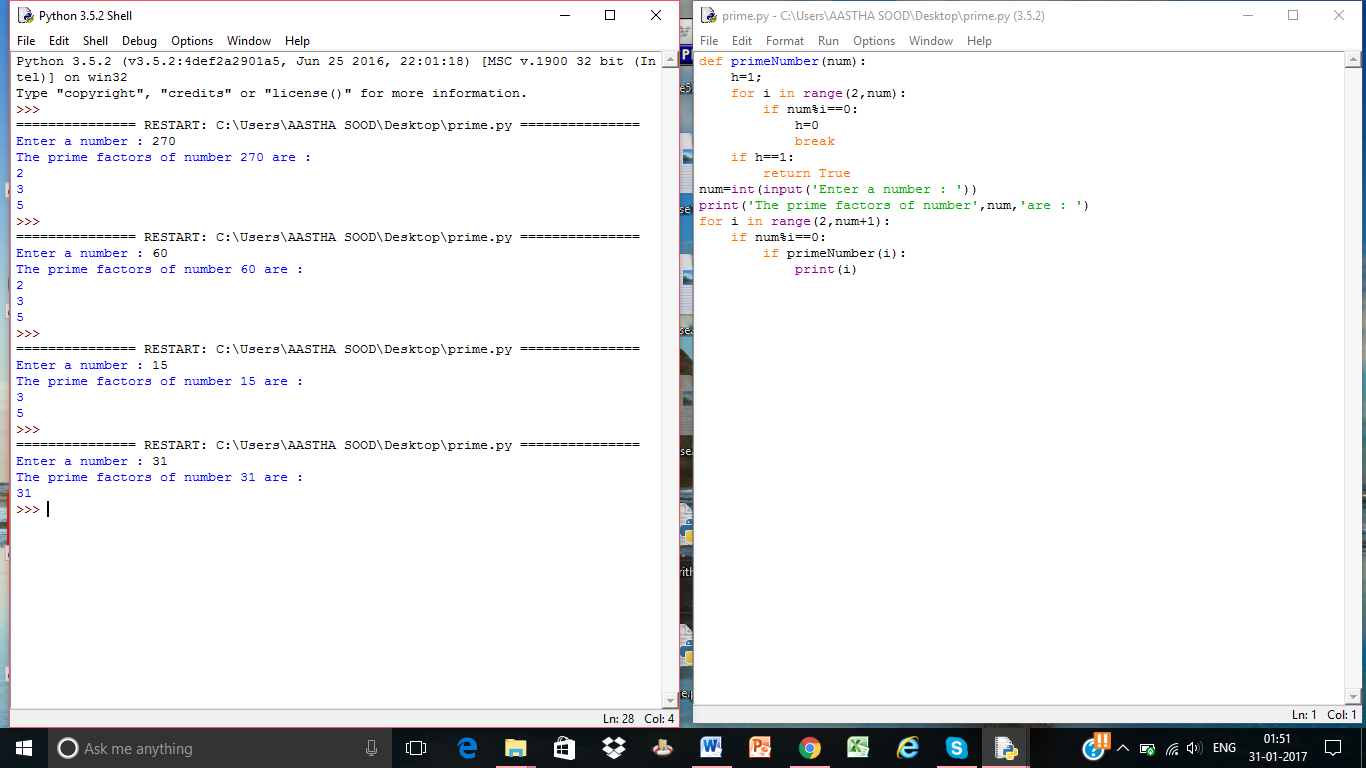
for i in range(2,num+1):

if num%i==0:

if primeNumber(i):

print(i)

**Program and output screenshot :**



Q2. THE 3X + 1 PROBLEM

Consider the following procedure to generate a sequence of numbers. Start with a positive integer n. If n is even, divide by 2. If n is odd, multiply by 3 and add 1. Repeat this process with the new value of n, terminating when n = 1. For example, the following sequence of numbers will be generated for n = 22:

22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

It is conjectured (but not yet proven) that this procedure will terminate at n = 1 for every integer n. For an input n, the cycle-length of n is the number of numbers generated up to and including the 1. In the example above, the cycle length of 22 is 16. Given any two numbers i and j, you must determine the maximum cycle length over all numbers between i and j, including both endpoints. Implement your code using Python.

**Algorithm:**

Input **Upper\_Limit**, **Lower\_Limit**

Create Empty List **k**

For **i** is **Lower\_Limit** in **Upper\_Limit**

Create Empty List **l**

**j**<-**i**

While ( **i** Not 1)

If (**i** Mod 2 is 0)

**i**=**i**/2

Append **i** to **l**

Else

**i**<-3\***i** +1

Append **i** to **l**

Append Length (**l**)+1 to **k**

Print Cycle length of **k** = Length (**l)**+1

Print Maximum cycle length = Max (**k**)

Python Code:

a = int(input('Endpoint 1 : '))

b = int(input('Endpoint 2 : '))

k=[]

for i in range (a,b+1):

l=[]

j = int(i)

while(i!=l):

if i%2==0:

i=i/2

l.append(int(i))

else:

i=3\*i+1

l.append(int(i))

k.append(len(l)+1)

print("Cycle Length of",j,"is",len(l)+1);

print("Maximum cycle length is",max(k))

Screenshot of program and output:

