

# Assignment Number

1

## Problem Statement

Program in C to print the prime factors of a number in descending order

## Theory

Factors of a number are the integers smaller or equal to the number itself, by which the number is completely divisible. A number may have any number of factors, but there is a certain range all of its factors must lie within. All integers are divisible by 1, so the least divisor of a number except for 1 should be 2. Let us consider a number N, which is divisible by 2. i.e.

$$N = 2 * M$$

Now since the result of multiplication ( $2 * M$ ) is constant, if we increase 2, M will decrease. Hence the largest factor of the number except the number itself is M. So the factors of any number except for 1 and the number itself will lie within the range  $[2, N/2]$ . A number which does not have any factors in the said range is called a prime number. Consequently, the factors of a number which themselves are prime numbers, are said to be the prime factors of the given number.

**Example :** 15 has 4 factors – 1, 3, 5, 15 – out of which 3 and 5 are only divisible by 1 and the number itself. Hence, they are prime factors of 15.

# Algorithm

**Input :** The number to search prime factors of, say N.

**Output :** The prime factors of the number in descending order, if any.

## Steps :

```

Step 1 : Print "Prime factors of " N " are : "
Step 2 : Repeat through step 4 to step 13 for i = N/2 to 2
Step 3 :   If(N mod i = 0)
           Then
Step 4 :       Set temp = i
Step 5 :       Set isprime = 1           // The prime flag
Step 6 :       Repeat through step 8 to step 10 for j = 2 to temp/2
Step 7 :           If(temp mod j = 0)
                   Then
Step 8 :               Set isprime = 0
Step 9 :               Goto step 11
                   [End of inner if structure]
Step 10 :           Set j = j + 1
                   [End of inner for loop]
Step 11 :       If(isprime = 1)
                   Then
Step 12 :           Print temp
                   [End of inner if structure]
                   [End of outer if structure]
Step 13 :       Set i = i - 1
                   [End of outer for loop]
Step 14 : End

```

# Source Code

```
#include <stdio.h>

int main(){
    int a, i, j, temp, isprime;
    printf("\nEnter the number : ");
    scanf("%d", &a); // Input the number
    printf("\nThe prime factors of %d are :", a);
    for(i = a/2; i >= 2; i--){ // Search for factors of `a`
        if(a % i == 0){ // `i` is a factor of `a`
            temp = i; // Store it to a temporary variable
            isprime = 1; // prime flag
            for(j = 2; j <= temp/2; j++){ // Search for factors of
                // `temp`
                if(temp % j == 0){ // Factor of `temp` is found
                    isprime = 0; // `temp` is not prime
                    break;
                }
            }
            if(isprime) // `temp` is prime
                printf(" %d", i);
        }
    }
    return 0;
}
```

# Input and Output

## Set 1 :

Enter the number : 12345

The prime factors of 12345 are : 823 5 3

## Set 2 :

Enter the number : 500000

The prime factors of 500000 are : 5 2

# Discussion

This program demonstrates a very basic approach towards the finding of prime factors of a given number, but it performs very poorly for large numbers. For example, this program makes a total of  $(N/2 - 2) * (P/2 - 2)$  iterations for a number  $N$  with  $P$  factors in the worst case. So for a sufficiently large number with a handful of factors can make this program run for quite a while. It can also be shown that if a number is constituted by multiplying two sufficiently large, random prime numbers, factorizing the resultant number is computationally infeasible with the resource presently we have at hand – which serves the basis of all cryptographic security services at present.