

## Efficient program to print all prime factors of a given number

Given a number  $n$ , write an efficient function to print all **prime factors** of  $n$ . For example, if the input number is 12, then output should be "2 2 3". And if the input number is 315, then output should be "3 3 5 7".

Following are the steps to find all prime factors.

- 1) While  $n$  is divisible by 2, print 2 and divide  $n$  by 2.
- 2) After step 1,  $n$  must be odd. Now start a loop from  $i = 3$  to square root of  $n$ . While  $i$  divides  $n$ , print  $i$  and divide  $n$  by  $i$ , increment  $i$  by 2 and continue.
- 3) If  $n$  is a prime number and is greater than 2, then  $n$  will not become 1 by above two steps. So print  $n$  if it is greater than 2.

```
// Program to print all prime factors
# include <stdio.h>
# include <math.h>
```

```
// A function to print all prime factors of a given number n
void primeFactors(int n)
{
    // Print the number of 2s that divide n
    while (n%2 == 0)
    {
        printf("%d ", 2);
        n = n/2;
    }

    // n must be odd at this point. So we can skip one element (Note
    for (int i = 3; i <= sqrt(n); i = i+2)
    {
        // While i divides n, print i and divide n
```

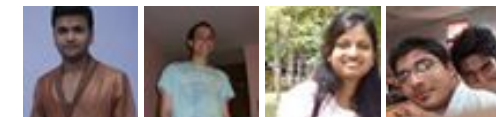
Google™ Custom Search



GeeksforGeeks



53,525 people like [GeeksforGeeks](#).



Interview Experiences

Advanced Data Structures

Dynamic Programming

Greedy Algorithms

Backtracking

Pattern Searching

Divide & Conquer

Mathematical Algorithms

Recursion

Geometric Algorithms

```

while (n%i == 0)
{
    printf("%d ", i);
    n = n/i;
}

// This condition is to handle the case when n is a prime number
// greater than 2
if (n > 2)
    printf ("%d ", n);
}

/* Driver program to test above function */
int main()
{
    int n = 315;
    primeFactors(n);
    return 0;
}

```

Output:

3 3 5 7

### How does this work?

The steps 1 and 2 take care of composite numbers and step 3 takes care of prime numbers. To prove that the complete algorithm works, we need to prove that steps 1 and 2 actually take care of composite numbers. This is clear that step 1 takes care of even numbers. And after step 1, all remaining prime factor must be odd (difference of two prime factors must be at least 2), this explains why  $i$  is incremented by 2.

Now the main part is, the loop runs till square root of  $n$  not till. To prove that this optimization works, let us consider the following property of composite numbers.

*Every composite number has at least one prime factor less than or equal to square root of itself.*

This property can be proved using counter statement. Let  $a$  and  $b$  be two factors of  $n$  such that  $a \cdot b = n$ . If both are greater than  $\sqrt{n}$ , then  $a \cdot b > \sqrt{n} * \sqrt{n}$  which contradicts the expression " $a * b = n$ ".

In step 2 of the above algorithm, we run a loop and do following in loop

- Find the least prime factor  $i$  (must be less than  $\sqrt{n}$ )
- Remove all occurrences  $i$  from  $n$  by repeatedly dividing  $n$  by  $i$ .

# Deploy Early. Deploy Often.

DevOps from Rackspace:

## Automation

[FIND OUT HOW ▶](#)



the open cloud company

### Popular Posts

[All permutations of a given string](#)

[Memory Layout of C Programs](#)

[Understanding "extern" keyword in C](#)

[Median of two sorted arrays](#)

[Tree traversal without recursion and without stack!](#)

[Structure Member Alignment, Padding and](#)

[Data Packing](#)

[Intersection point of two Linked Lists](#)

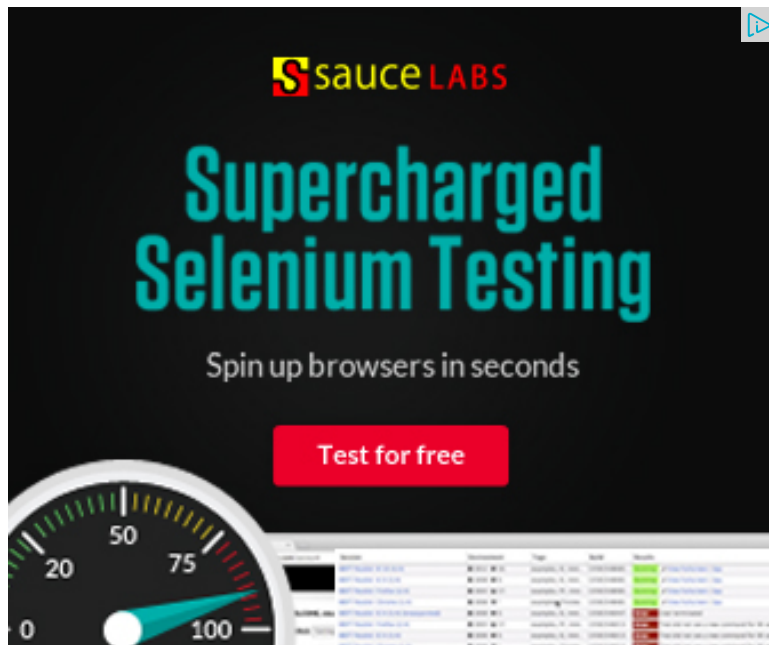
[Lowest Common Ancestor in a BST.](#)

[Check if a binary tree is BST or not](#)

[Sorted Linked List to Balanced BST](#)

c) Repeat steps a and b for divided n and  $i = i + 2$ . The steps a and b are repeated till n becomes either 1 or a prime number.

Thanks to **Vishwas Garg** for suggesting the above algorithm. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above



## Related Topics:

- [Backtracking | Set 8 \(Solving Cryptarithmic Puzzles\)](#)
- [Tail Recursion](#)
- [Find if two rectangles overlap](#)
- [Analysis of Algorithm | Set 4 \(Solving Recurrences\)](#)
- [Print all possible paths from top left to bottom right of a mXn matrix](#)
- [Generate all unique partitions of an integer](#)
- [Russian Peasant Multiplication](#)
- [Closest Pair of Points | O\(nlogn\) Implementation](#)



40



Tweet

4



3

Writing code in comment? Please use [ideone.com](#) and share the link here.

## HP Chromebook 11



[google.com/chromebook](https://google.com/chromebook)

Everything you need in one laptop. Made with Google. Learn more.

[Programming Xcode](#)

[Informix Database](#)

[Free C# Code Generator](#)

[NuoDB: The Elastic SQL DB](#)

[Learn Xcode and iOS](#)

[Html5 Tutorial](#)

[Free Downloadable](#)



## Recent Comments

Abhi You live US or India?

[Google \(Mountain View\) interview](#) · 12 minutes ago

**Aman** Hi, Why arent we checking for conditions...

[Write a C program to Delete a Tree.](#) · 52 minutes ago

kzs please provide solution for the problem...

[Backtracking | Set 2 \(Rat in a Maze\)](#) · 56 minutes ago

**Sanjay Agarwal** bool

tree::Root\_to\_leaf\_path\_given\_sum(tree...


[Root to leaf path sum equal to a given number](#) · 1 hour ago

**GOPI GOPINATH** @admin Highlight this sentence "We can easily...

[Count trailing zeroes in factorial of a number](#) · 1 hour ago

**newCoder3006** If the array contains negative numbers also. We...

[Find subarray with given sum](#) · 1 hour ago

AdChoices 

[▶ C++ Code](#)

[▶ Programming C++](#)


[▶ Prime Number](#)

AdChoices 

► [Print a Number](#)

► [Math Number](#)

► [Code Number](#)

AdChoices 

► [Numbers Number](#)

► [C++ Java](#)

► [C++ Program](#)

@geeksforgeeks, **Some rights reserved**

**Contact Us!**

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team