# C# Advanced Lab - Algorithms

This document defines **algorithmic problems** from the ["Advanced C#" Course @ Software University](http://softuni.bg/courses/advanced-csharp/). You are presented with some problems and certain steps you need to take in order to accomplish the tasks.

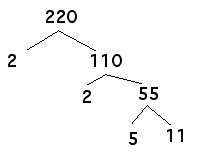
## Prime Factorization

*Fun fact: did you know int.MaxValue (231 – 1) is a prime?*

Prime factorization of a number N is the process of finding a set of prime numbers that multiply together to produce N. E.g. 12 can be represented as 2 \* 2 \* 3; 534543 = 3 \* 23 \* 61 \* 127.

There are useful online calculators you can use to check the prime factorization of a number, like [this one](http://www.calculatorsoup.com/calculators/math/prime-factors.php).

**The task**: Write a program that takes as input an **integer number N (N >= 2)** and represents it as a multiple of prime numbers in format: **"[number] = [prime factor 1] \* [prime factor 2] \* … \* [prime factor n]"**.

Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 | 2 = 2 |
| 12 | 12 = 2 \* 2 \* 3 |
| 220 | 220 = 2 \* 2 \* 5 \* 11 |
| 534543 | 534543 = 3 \* 23 \* 61 \* 127 |

One **possible** approach:

1. Create a **list** to hold each prime multiple.
2. Set a variable **divisor** to 2 (the first prime number).
3. Check if N can be divided by divisor:
   1. If you can divide N by divisor without remainder, add divisor to the list and divide N by divisor. Repeat this step.
   2. If you cannot divide N by divisor without remainder, increment divisor and repeat step 3.
4. End the process when N equals 1.
5. Print the result in the specified format.

### Restrictions

* The number N will always be a positive integer in the range [2 … 2 000 000 000]. There is no need to check it explicitly.
* The prime factors of the number should be sorted in ascending order.
* Allowed working time for your program: 0.9 seconds.
* Allowed memory: 16 MB.