# Listing Factors

## Factors

The **factors** of a number are any numbers that divide into it exactly. This includes 1 and the number itself.

For example, the factors of 6 are 1, 2, 3 and 6. The factors of 8 are 1, 2, 4 and 8.

The program that you will write will output the list of factors for a number that the user types in at the console. Each step will take you closer to that goal.

## Stage 1 – User Enters A Number

 Dim usernumber As ?

 Console.Write("Enter a number: ")

 usernumber = ?

 Console.WriteLine("Factors of {0}.", ?)

 Console.ReadLine()

1. Replace the question marks shown here with the words that are needed to make the program work. Make a new program using this code.
2. Take a screenshot of your programming running. Include it in your work and explain how it shows that your program works correctly.

## Stage 2 – The Obvious

1. All whole numbers divide by 1 and give no remainder. We don’t need our program to work anything out to do this. Write a line of code to output the number 1.
2. The number itself is also a factor. Write another line of code to output this number on the line below the 1.
3. Test your program.

## Stage 3 – The Other Possible Factors

To work out the factors of a number, we first write down the number 1 because we know it’s a factor. Then we see if the number can be divided by 2 without a remainder. If it can, we write that down, then we try 3, then 4 and so on. There has to be a stopping point. Consider the following facts,

* The smallest factor of any number will always be 1.
* The largest factor of any number is always the number itself.

Take the number 12.

* The smallest factor of 12 is 1.
* The largest factor is 12.
* 12 divided by 1 = 12.
* 12 divided by 12 is 1.

Now think about the second smallest and second largest factors.

* The smallest **second smallest** *possible* factor a number can have is 2. This happens only when the number is even.
* If we divide a number by 2, we get exactly half of the number.
* The **second largest factor** cannot be larger than half of the number.

This will be helpful with code,

Make space between the lines of code you wrote to output the 1 and the number itself. This is where you will write the code that finds the factors,

 For check As Integer = 2 To Math.Floor(usernumber / 2)

 Next

1. Enter these two lines. In between the two lines, write an output statement. It should say **Possible Factor:** followed by the value currently stored by the loop counter variable.
2. Test this program.

## Stage 4 – Checking Divisibility

In the last stage, you wrote a line of code that displayed the possible factors of a number. In this stage, you need to check if that number is actually a factor and only output it **if** it is.

You are going to need an IF statement.

You know that a number is a factor of another if it divides into that number without a remainder. Visual Basic has an operator for that. Consider the following,

|  |  |
| --- | --- |
| 5 Mod 2 | 1 |
| 4 Mod 2 | 0 |
| 9 Mod 3 | 0 |
| 9 Mod 2 | 1 |
| 9 Mod 5 | 4 |
| 9 Mod 6 | 3 |

Each time you are getting the remainder when the first number is divided by the second. If that remainder is a 0, the second number is a factor of the first number.

If you get a remainder of 0 when you divide the usernumber by check, if usernumber Mod check is equal to 0 then, the number is a factor and should be displayed on the screen.

1. Read the explanation carefully. Replace the line that displays a possible factor with an IF statement. If usernumber Mod check is equal to 0 then, the number is a factor and should be displayed on the screen.
2. Test your program. You should now see all of the factors of a number. Include some screenshots and an explanation of how you know that your program works correctly.

## Extension 1 – Perfection

There is a special sequence of numbers in Maths called **perfect numbers**.

A perfect number is the **sum** of its **proper divisors**.

The proper divisors of a number are all of its factors **except** the number itself.

1. Declare an integer variable at the start of your program and call it **total**. Make it equal to 1, just before the loop that checks all the possible factors. Each time you find a factor add it to the total (total = total+check). After you output the number itself, add an IF statement. If the total is equal to the usernumber, say that the number is a **perfect number**.
2. Test your program. Look up **perfect numbers** online and see if you can find enough information to test your program properly.

## Extension 2 – One More Challenge

If you felt that this task was a little on the easy side for you, take on one last challenge. Work out how to make your program output all of the perfect numbers below 9,223,372,036,854,775,807. You will need to change your program so that the integer variables are now of the data type **long**. This includes the loops. This is necessary to allow you to explore larger numbers. Your program will take a wee while to run. If it goes very slowly, remove some digits from the right of the big number.