Happy Numbers

# Introduction

To find out if a number is a **happy number**, we carry out the following steps.

Take each digit in the number and square it, adding up the sum of all those squares.

Keep repeating the process with the result until the result is either 1 (which means that the number you started with was a happy number or you keep getting a repeated cycle of numbers.

For example,

|  |  |  |
| --- | --- | --- |
| **Number** | 49 |  |
| Sum of squares of digits | 42 + 92 = 16 + 81 = | 97 |
| Sum of squares of digits | 92 + 72 = 81 + 49 = | 130 |
| Sum of squares of digits | 12 + 32 + 02 = 1 + 9 + 0 = | 10 |
| Sum of squares of digits | 12 + 02 = 1 + 0 = | **1** |
| **Result** | 49 is a happy number |  |

Another example,

|  |  |  |
| --- | --- | --- |
| **Number** | 11 |  |
| Sum of squares of digits | 12 + 12 = 1 + 1 = | 2 |
| Sum of squares of digits | 22 = | **4** |
| Sum of squares of digits | 42 = | 16 |
| Sum of squares of digits | 12 + 62 = 1 + 36 = | 37 |
| Sum of squares of digits | 32 + 72 = 9 + 49 = | 58 |
| Sum of squares of digits | 52 + 82 = 25 + 64 = | 89 |
| Sum of squares of digits | 82 + 92 = 64 + 81 = | 145 |
| Sum of squares of digits | 12 + 42 + 52 = 1 + 16 + 25 = | 42 |
| Sum of squares of digits | 42 + 22 = 16 + 4 = | 20 |
| Sum of squares of digits | 22 + 02 = 4 + 0 = | **4** |
| **Result** | 11 is **not** a happy number |  |

In this second example, the sequence returns to 4. If the sum of the squares of the digits is ever one of these numbers, the number is not happy. For all unhappy numbers, this sequence will be reached somehow. Checking for a result of 4 is enough to determine that a number is not happy.

# Task 1 – Understanding Happiness

Before starting any of the tasks, make sure you are totally confident with the concept.

Test the following by hand and determine if they are or are not happy numbers.

1. 44
2. 21
3. 133

# Task 2 – Digits

Look back at the examples on the previous page. Part of this program is going to need you to be able to look at each digit of a number.

There are a few ways to do this, but the best way to do it is to think of it mathematically. The following flowchart describes the process for you.



You can see a loop in this part of the flowchart. It is based on a question asked before deciding whether or not to repeat some statements.

The loop stops when the value of n reaches 0.

You will need to use a **while** loop to achieve this.

The format you use for a while loop is as follows,

 While **condtion**

 End While

You need to write the condition where it is indicated. The code that is to be repeated goes **in between** these lines.

The output statement includes the modulus or remainder operator, MOD.

In the statement that comes after it, the integer division operator DIV is used. In VB, the integer division operator is a backslash.

1. Use the flowchart to create the code that outputs all of the digits of a given number. Test this first stage.
2. Explain, in your own words, how this algorithm does its job.

# Task 3 – Sum Of Squares Of Digits

A small addition to the code that you wrote for the first stage will calculate the sum that you need.

 Dim sum As Integer = 0

 sum = 0

 While ?

 sum = sum + ?

 n = ?

 End While

Look at the section of code written above. It should look like your code with a few additions. There is the variable **sum,** which is declared and assigned the value 0. There are question marks in the **while** loop and the line that assigns a value to **n**. These are things that you should have written in the first stage.

Make sure that you put the variable declaration at the top of the code. You will be making sum equal to 0 each time you calculate the sum of the squares of the digits.

1. Adapt your code from the last stage so that it looks like the sample. The key line for you to work out is how to set the value of the variable **sum**, the others should still be right. Instead of outputting the digit, n mod 10, you need to multiply it by itself (square it) and add it to the sum. Output the sum at the end.
2. Test this and include some evidence of your tests in your documentation.

# Task 4 – Working Out If A Number Is Happy

A flowchart to describe this process is,

Notice that one of processes is the outcome of the code you have written so far. It is made equal to n, the variable used to store the number that was input.

The first diamond decision box represents a while loop.

The second decision represents an IF statement.

1. In the flowchart, the code that makes the sum of the squares of the digits is part of the repetition loop (the while loop you will need to write). The first diamond shows the condition you will need. When you have calculated the sum of the squares of the digits, make n equal the sum so that the process can be repeated if the result is not 1 or 4. Write the necessary IF statement after the loop to output whether or not the number is happy.

# Task 5 – Use It Or Lose It

1. Use a for loop to output all of the happy numbers from 1 to 1000. Look up the Wikipedia entry for happy numbers and compare your list.

# Task 6 – Extension – Happy Primes

A happy prime is a happy prime which is also a prime number. Extend the program so that it finds only prime happy numbers. You will be able to find a list of some of these online to see if you are correct.