Happy numbers

Let the sum of the squares of the digits of a positive integer s_0 be represented by s_1. In a similar way, let the sum of the squares of the digits of s_1 be represented by s_2, and so on.

Iterating this sum-of-squared-digits map always eventually reaches one of the 10 numbers 0, 1, 4, 16, 20, 37, 42, 58, 89, or 145.

If s_i=1 for some i>=1, then the original integer s_0 is said to be happy. For example, starting with 7 gives the sequence 7, 49, 97, 130, 10, 1, so 7 is a happy number.

The first few happy numbers are 1, 7, 10, 13, 19, 23, 28, 31, 32, 44, 49, 68, 70, 79, 82, 86, 91, 94, 97, 100, ... These are also the numbers whose 2-recurring digital invariant sequences have period 1. The numbers of iterations required for these to reach 1 are 0, 5, 1, 2, 4, 3, 3, 2, 3, 4, 4, 2, 5, ...

The numbers of happy numbers less than or equal to 1, 10^1, 10^2, ... are given by 1, 3, 20, 143, 1442, 14377, 143071, ...

The first few consecutive happy numbers (n,n+1) have n=31, 129, 192, 262, 301, 319, 367, 391, ... Similarly, the first few happy triplets start with 1880, 4780, 4870, 7480, 7839, ...

The first few happy primes are 7, 13, 19, 23, 31, 79, 97, 103, 109, 139, ...

Once it is known whether a number is happy (or not), then any number in the sequence s_1, s_2, s_3, ... will also be happy (or not). A number that is not happy is called unhappy. Unhappy numbers have eventually periodic sequences of s_i which never reach 1.

Any permutation of the digits of an unhappy or happy number must also be unhappy or happy. This follows from the fact that addition is commutative.

http://mathworld.wolfram.com/HappyNumber.html

Goal of these exercises

The aim of these exercises is to find the happy numbers up to 1 million and study their properties.

In these exercises, we will find the happy numbers using ROOT classes and methods and we will construct and save some variables on a ROOT file. Then we will use a script to plot the happy numbers and we will try to understand if their density is changing with increasing values.

Exercise 1

Write an an executable compiled program which loops from 1 to 1e6 (one million) and check for each integer if it is a happy number or not.

The program must give as output a ROOT file containing a TTree with the following three variables (a TBranch for each one):

- number : the integer
- happy : an integer number, 1 if “number” is happy, 0 otherwise
- happycount : an integer number that is increased every time a happy number is found.

NB: save in the output file all the 1e6 numbers, not only the happy ones.

Important hints:

- Use a fixed length array to store the sequence needed to check if 1 or a cycle is reached; the dimension of this array can be 1000.
- Make use of TString class, i.e. convert the integer number into a TString (remember: if `TString s=”ciao”` then `s[0]=”c”`, `s[1]=”i”`, etc.).
- Look at methods: TString::Sizeof(), TString::Atoi().
- Make use of a while-loop to check for “happiness” and go through the sequence and of a for-loop to calculate the square sum at each step.

Exercise 2

Write a ROOT-CINT script which reads the output file of exercise 2 (should be similar to the provided one use this file if you are unable to complete or run exercise 1), selects the happy numbers only and gives as output on the screen and on the disk (pdf format) a TCanvas, divided into two pads (one column, two rows – hint: `TCanvas::Divide`), which contains from top to bottom:

1. A plot (use TGraph) of the happy numbers versus their order of appearance , i.e. on the Y axis the happy numbers on X axis the related happycount variable (hints: there are less than 150000 happy numbers below one million, use fixed length arrays);
2. the distribution of happy numbers, (use `TH1D,`X range [0., 1e6], number of bins: 100).

Fit the distribution of happy numbers (point 2) with a straight line ("pol1") - set all weights to one for non empty bins, ignore error bars (hints: look at reference guide, TH1 class). Print on the STDOUT the parameters obtained from the fit. Is the density of happy number constant, increasing or decreasing in this range?

Preparing the output

- create a directory and put inside this directory ALL the files you want me to correct and look at.
- create a README text file (named like EM_README.txt), inside the file write:
  - **your name and surname**
  - a list of the files you are submitting
  - **in details** how to compile and run the programs
  - any other comment and answer to question(s)

Timing and rules

- You have four hours time to do your work.
- You can search the web, look at manuals, look at any note you wrote during the course, etc.