Practical Example: Hamming distance

Assignment 1 Hamming distance of two DNA sequences (10)

In this assignment you will implement an algorithm in Matlab for computing the Hamming distance of two DNA sequences and displaying the matching positions.

a. Study the file `hamming.m` in the subdirectory `input`. This is a so-called Matlab M-file. The appendix on page 3 gives a listing of this M-file. Study the code and the comments, these should be self-evident. If necessary, consult the Matlab tutorial of the first practical, or the `help` function of Matlab itself.

b. Load the file `hamming.m` in an editor, and add the missing code. The program should compute a number HD that equals the Hamming distance of two strings `s` and `t`, and print this number HD on the screen. Call your program `hamming1.m`.

c. Run your program `hamming1.m` in Matlab. It will use the file `input.txt` as input. Check that your program gives the correct answer for the strings in the input file.

d. Extend your program so that it displays the input strings below one another, with a line in between which has a '|' symbol on position `k` if a match occurs between the input strings at that position, and a space otherwise, like in the following example:

$$\begin{array}{cccc} C & A & G & G \\ | & | & & | \\ C & A & T & G \end{array}$$

Call your program `hamming2.m`.

e. Run your program `hamming2.m`, which again has `input.txt` as input. Check that your program displays the correct answer on the screen for the strings in the input file.

f. Instead of printing the results on the screen by the `disp` or `printf` function you can also print the results to a file by the `fprintf` function, as follows. Include the following lines[1] at the end of your program `hamming2.m`, and add the missing code for printing. The code for printing the value of the Hamming distance has already been inserted. Consult Matlab's `help` function about printing (strings) of characters.

```
% Print the alignment to a file:
output=fopen('hamming3-output.txt', 'w');      % open file
fprintf(output,'Name: <Your name(s)>\n');      % enter your name(s)
fprintf(output,'IBC, Practical Example\n');

fprintf(output,'Hamming distance=%d\n',HD);
fprintf(output,'\nAlignment:\n\n');
% Here comes the code for printing the alignment
....

fclose(output);                                % close file
```

[1]These can also be found in the file `print_to_file.m` in the subdirectory `input`.

Call your extended program `hamming3.m`. Run it again and check that the output file `hamming3-output.txt` contains the correct information.

Hand in:

- a concise report in PDF format (preferably generated by LaTeX), in which:

  a. you describe, for each part of every assignment, how you arrived at the solution,
  b. you answer all the questions posed in the assignments.

  Include the file `hamming3-output.txt` also in your report. In LaTeX this can easily be done by using the command `\verbatiminput{hamming3-output.txt}` (you need to include `\usepackage{verbatim}` in the preamble of your LaTeX document).

  Write at the top of the first page of your report: "**Introduction to Scientific Computing, Practical Example**", followed by your name(s), student number(s), and the date when you hand in the report.

- the files `hamming1.m`, `hamming2.m`, `hamming3.m` containing your final implementations.

- The file `hamming3-output.txt` containing the Hamming distance and alignment of the input strings.

All files have to be handed in as **a single archive** (called `YourName.zip` or `YourName.tgz`), where "YourName" is the concatenation of your last names (or your last name, if you work individually). See Nestor for the address to which you have to send the archive.

Appendix: the skeleton program `hamming.m`

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Introduction to Scientific Computing - WBCS14003              %
%                                                               %
%  Compute the Hamming distance for 2 sequences in Matlab       %
%                                                               %
%                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;                           % remove items from the workspace

in=fopen('input.txt');               % open file
s=fgetl(in);                         % read line 1 of the input
t=fgetl(in);                         % read line 2 of the input
fclose(in);                          % close file
len_s=length(s);                     % length string s
len_t=length(t);                     % length string t

% Here comes your code
.................
.................
```