Numerical Methods (NUM101) — Coursework 1

This is the first of three courseworks which combine for 50% of the credits for this unit. The maximum number of marks for this coursework is 10, worth 16% of the overall credits. This coursework consists of **one** question.

Instructions and rules

- Deadline: 22 March 2010.
- Material to be handed in:
  - (**Hardcopy**) Printed document containing printout of test function `TestMyAdd.m` (see below for instructions) and comments (no essay!),
  - (**Hardcopy**) printout of program codes with comments.
  - (**Victory**) Upload the `m`-file `MyAdd.m` to the Victory assignment `CW1`.

  The working function `MyAdd` accounts for 100% of the credits. The hardcopy is only there to show the output, the code, and give additional comments.
- Credit for this question:

  100% code performs computation correctly and efficiently, is well structured and commented;

  ≥80% code performs computation correctly and efficiently

  ≥60% code performs computation correctly but has problems[1];

  ≥40% code does not perform computations correctly but could be made to work with minor corrections;

  ≥20% the intentions behind the code are discernible with some effort.
- This is **individual** coursework. If you copy code from other students you will be turned in for plagiarising.
- For questions, clarifications and further help contact:

  Jan Sieber (`jan.sieber@port.ac.uk`, office LG.146).

[1] for example, the function works correctly most of the time but fails for some valid arguments

Question 1: A function to add large numbers

Matlab can only handle integer numbers up to 2^{53} ($\sim\ 10^{16}$) accurately. But we can use row vectors such as `x=[1,9,8]` or `y=[2,0,4,9]` that consist only of single digit numbers from `0` to `9` to represent arbitrarily large non-negative integers (in this case `x=198`, `y=2049`). Write a function that takes two row vectors of single digits and adds them up, treating the row vectors as integers. This coursework is not directly related to *Numerics* but it gives you the opportuniy to practise your generic programming skills. [10 marks]

Step-by-step instructions:

1. Create a new `m` file and save it as `MyAdd.m`. This file will contain the function `MyAdd`.
2. The first line of the function file `MyAdd.m` has to look like this:

   ```
   function res=MyAdd(x,y)
   ```

   Add your code and test your function on some examples (see Hints).
3. Download the file `TestMyAdd.m` from Victory (see Hints) into your current directory. Execute

   ```
   TestMyAdd(123456)
   ```

   on the command-line where you **replace** `123456` by your HEMIS number. `TestMyAdd` will first test your function on several example inputs and print out results or error messages. Then it will output some magic number. The output of `TestMyAdd` has to be submitted as hardcopy.

Total for Question 1: 10 marks

Hints:

- Your function should behave like this on the command-line:

```
>> s=MyAdd([1 8 9],[2 0 4 9])
s =
     2     2     3     8
>> MyAdd(9*ones(1,40),7*ones(1,39))
ans =
  Columns 1 through 13
     1     0     7     7     7     7     7     7     7     7     7     7     7
  Columns 14 through 26
     7     7     7     7     7     7     7     7     7     7     7     7     7
  Columns 27 through 39
     7     7     7     7     7     7     7     7     7     7     7     7     7
  Columns 40 through 41
     7     6
```

- You do not have to worry about negative numbers. You can assume that all input vectors consist only of single digits and that the first digit is greater than zero. (This means that you do not need to check the input arguments for sanity.)
- The function should return a vector that contains only non-negative single digit numbers. So `[2,24,7]` instead of `[2,2,4,7]` is not valid.
- Remove leading zeros from the result: `[0,2,2,4,7]` is not valid.

Question 1 continues overleaf

- On Victory in folder `Coursework 1` you will find the file `TestMyAdd.m`. Download this file into your current Matlab working directory.
- Make sure that all statements in your function `MyAdd` are quiet (finish with a semicolon). Otherwise, this may mess up the output of the test script `TestMyAdd`
- Beware that the Matlab installation in the Lab may have an annoying bug and claim that it cannot find `TestMyAdd.m`, or `MyAdd.m`, even though it is in the current working directory (check by typing `ls`).

  You can work around this bug by working on a directory on a USB stick instead of the `N:` drive. If you insist on working on `N:` drive:

  – open `TestMyAdd.m` into the editor,
  – click on the `Run` toolbar button in the editor,
  – a pop-up window may show up (click on `Change directory`),
  – ignore the error message on the command-line,
  – try again `TestMyAdd([2,3,4,5,6,7])` on the command-line to see if you still get an error message claiming that `TestMyAdd` is not found.

  You may have to do this also for all your newly written or downloaded functions and scripts in your current directory. I am in contact with IS to sort this out.