Communications Network Design
Class Exercise 1: before lecture, Thursday, April 2nd, 2009

Matthew Roughan
<matthew.roughan@adelaide.edu.au>

Write a version of Dijkstra's algorithm in Matlab, carefully following the instructions below.

1. The solution should be in the form of a function, in a ".m" file, with the name of the file being:

   ```
   dijkstra_N.m
   ```

   where N is your student number.

2. The function should be called as follows:

   ```
   [D, P] = dijkstra_N(A, s);
   ```

   where the inputs are:

   - `A` = the connectivity matrix of the input graph, i.e.

     $$A_{ij} = \begin{cases} d_{ij}, & \text{if node } i \text{ and } j \text{ are connected by an edge,} \\ \infty, & \text{otherwise .} \end{cases}$$

     Note that the matrix may be directed, and so `A` isn't necessarily symmetric. In matlab, the term `A(i,j)`, i.e., the element at row i, and column j refers to the weight of link from node i *to* node j. A link weight of ∞ means there is no link. The diagonal terms should be zero.
   - `s` is the starting node, from which to find shortest paths to all other nodes.

   and the outputs are:

   - `D` = a vector giving the distance from `s` to each node in the network (i.e. D_i = distance from `s` to `i`).
   - `P` = a vector giving the predecessor nodes in the SPF tree. Note that the predecessor of node `s` should be NaN.

   All vectors are to be entered and output as column vectors.

3. Your exercise will be checked by automatically running it on a set of 10 *directed* graphs. You will receive one mark out of 10 for each network on which your code works correctly. If your code doesn't work, or doesn't fit the above form, you will get **zero**.

   - It is possible that the graph may not be connected, i.e., there may be no possible path from s to i. In this case, the correct output is $D_i = \infty$, and $P_i = NaN$.
   - There are several possible errors you might detect if input data is incorrect, the following is a list of those I might test, and the correct output:
     - Weights mus tbe non-negative. If not, output `error: weights must be non-negative!`
     - `A` must be a square matrix. If not, output `error: A must be square!`
     - Given N nodes (`A` is $N \times N$), then the start node must be in the set $s \in \{1, 2, \ldots, N\}$. If not, output `error: s out of range!`

4. If any help is required with Matlab, see me as soon as possible.

5. Your code should be standalone, in one file! No subroutines!.
6. Your code should produce NO outputs other than those requested. It should not print out any values. Please be careful about this. The scripts are automatically marked, and any extra outputs will cause the script to crash, and you will receive zero.
7. Test examples are available from the web page. These are not the examples I will use to validate your code, but if you can get the correct results for these, then you are likely to do well on the validation problems.
8. **Handin instructions:** We will use MyUni for handing up assignments. You need to make sure your matlab file has the right name, and then
   (a) Go to `Communications Network Design` on the MyUni webpage.
   (b) Select `Assignments` and then `Assignment 1`
   (c) Select the `HANDIN` link.
   (d) Enter a comment. MyUni requires one, but we don't read them, so it can be almost anything.
   (e) Attach the assignment file
   (f) Click 'Submit' - NOT Save.

   Note that you can only submit your assignment once. Make sure that you have done all that is required before continuing.