CSC 121, Spring 2017 — Small Assignment #2

Worth 5% of the course grade. Due by the start of class on February 14, to be handed in using MarkUs. This assignment may be handed in late, with a 20% penalty, by start of class on February 17. Assignments will not usually be accepted after that. Contact the instructor as soon as possible if you have a legitimate excuse (such as documented illness) for handing in the assignment late (without penalty).

This assignment is to be done by each student individually. You may discuss it in general terms with other students, but the work you hand in should be your own. In particular, you shouldn't leave a discussion with someone else with any written notes (either paper or electronic).

In this assignment, you will write and test an R function for finding the distances to a "target" location on a grid from all locations on this grid, measuring distance along the shortest path that avoids obstacles.

Something like this function (but probably more complicated) might, for example, be needed if you wanted to examine whether death rates from home accidents are related to how far the home is from the nearest hospital, or if you wanted to analyse whether the number of squirrels at different points in a wooded area is related to how far that point is from a pond where water is available.

We will consider locations on a rectangular grid, with obstacles indicated on a map, which is represented in R by a matrix of strings, which are either `" "`, meaning the location can be moved through freely, or `"X"`, meaning that the location is an obstacle, and movement into or out of this location is not possible. Locations will be identified by a vector of two integers, giving the row and column of the location in this map matrix.

The distance between two locations will be defined to be the smallest number of vertical or horizontal moves needed to get from one of these locations to the other, without ever being in a location that is marked `"X"` on the map. Diagonal moves are not considered. If their is no way of moving between the locations without passing through an obstacle, or if either location is itself located at an obstacle, the distance is infinite, written `Inf` in R.

To find all distances from a target location, you will write an R function call `find_distances`, whose definition should start as follows:

```
find_distances <- function (map, target)
```

The `map` argument should be a matrix of strings that both defines the extent of the grid and indicates where obstacles are, as described above. The `target` argument should be a vector of two numbers, giving the row and column coordinates for the "target" location. The value returned by this function should be a numeric matrix with the same number of rows and columns as `map`, giving the distance of each point on the grid from the target location. The target location itself will have distance zero. Some locations may have infinite distances, if it's not possible to get to the target from there.

Here are some tests showing what the `find_distances` function should do:

```
> map <- matrix (c (
+                  " ", "X", " ", " ",
+                  "X", " ", "X", " ",
+                  "X", " ", "X", " ",
+                  " ", " ", " ", " ",
+                  " ", " ", " ", " "),
+          nrow=5, ncol=4, byrow=TRUE)
```

```
> map
     [,1] [,2] [,3] [,4]
[1,] " "  "X"  " "  " "
[2,] "X"  " "  "X"  " "
[3,] "X"  " "  "X"  " "
[4,] " "  " "  " "  " "
[5,] " "  " "  " "  " "

> find_distances(map,c(1,4))
     [,1] [,2] [,3] [,4]
[1,]  Inf  Inf    1    0
[2,]  Inf    7  Inf    1
[3,]  Inf    6  Inf    2
[4,]    6    5    4    3
[5,]    7    6    5    4

> find_distances(map,c(3,2))
     [,1] [,2] [,3] [,4]
[1,]  Inf  Inf    7    6
[2,]  Inf    1  Inf    5
[3,]  Inf    0  Inf    4
[4,]    2    1    2    3
[5,]    3    2    3    4

> find_distances(map,c(2,3))
     [,1] [,2] [,3] [,4]
[1,]  Inf  Inf  Inf  Inf
[2,]  Inf  Inf    0  Inf
[3,]  Inf  Inf  Inf  Inf
[4,]  Inf  Inf  Inf  Inf
[5,]  Inf  Inf  Inf  Inf
```

Whe writing your function, you may assume without checking that the `map` and `location` arguments are valid.

You *must* indent your function definitions properly, as illustrated by the examples in the lecture slides. You should include any comments that are helpful in explaining what your code is doing, but you do not need to document what the function does, since that is documented already in this assignment handout.

You should hand in two script files, one with only the definition of your function, and the other with tests of this function, starting with the tests above (for which the script is available from the course web page), to which you should add some tests of your own. You should also hand in the output of your test script, as a text file.

Here is a suggested method for solving this problem. Start by creating a matrix of distances in which all distances are `Inf`, except that the distance for the target location is 0. Then go around a loop that each iteration scans all locations to see if the distance at that location can now be seen to be less than what it is currently recorded as, based on the distances for locations that are adjacent to it (vertically or horizontally). Stop this loop when the last iteration did not change the distance at any location.