You must implement a Matlab function or a Python executable file called value\_iteration. Your function should be invoked as follows:

value\_iteration(<environment\_file>, <non\_terminal\_reward>, <gamma>, <K>)

If you use Python, just convert the Matlab function arguments shown above to command-line arguments. The arguments provide to the program the following information:

* The first argument, <environment\_file>, is the path name of a file that describes the environment where the agent moves (see details below). The path name can specify any file stored on the local computer.
* The second argument, <non\_terminal\_reward>, specifies the reward of any state that is non-terminal.
* The third argument, <gamma>, specifies the value of γ that you should use in the utility formulas.
* The fourth argument, <K>, specifies the number of times that the main loop of the algorithm should be iterated. The initialization stage, where U[s] is set to 0 for every state s, does not count as an iteration. After the first iteration, if you implement the algorithm correctly, it should be the case that U[s]=R[s].

|  |
| --- |
| Shape  Description automatically generated  Figure 1: The environment described in file [environment1.txt](https://athitsos.utasites.cloud/courses/cse4309_fall2021/assignments/assignment7/environment1.txt). |

The environment file will follow the same format as files [environment1.txt](https://athitsos.utasites.cloud/courses/cse4309_fall2021/assignments/assignment7/environment1.txt) and [environment2.txt](https://athitsos.utasites.cloud/courses/cse4309_fall2021/assignments/assignment7/environment2.txt). For example, file [environment1.txt](https://athitsos.utasites.cloud/courses/cse4309_fall2021/assignments/assignment7/environment1.txt) describes the world shown in Figure 1, and it has the following contents:

1.0,X

.,-1.0

|  |
| --- |
| Logo  Description automatically generated Figure 2: The environment described in file [environment2.txt](https://athitsos.utasites.cloud/courses/cse4309_fall2021/assignments/assignment7/environment2.txt). |

Similarly, file [environment2.txt](https://athitsos.utasites.cloud/courses/cse4309_fall2021/assignments/assignment7/environment2.txt) describes the world shown in Figure 2, and it has the following contents:

.,.,.,1.0

.,X,.,-1.0

.,.,.,.

As you see from the two examples, the environment files are CSV (comma-separated values) files, where:

* Character '.' represents a non-terminal state.
* Character 'X' represents a blocked state, that cannot be reached from any other state. You can assume that blocked states have utlity value 0.
* Numbers represent the rewards of TERMINAL states. So, if the file contains a number at some position, it means that that position is a terminal state, and the number is the reward for reaching that state. These rewards are real numbers, they can have any value.

**Implementation Guidelines**

* For the state transition model (i.e., the probability of the outcome of each action at each state), use the model described in pages 9-10 of the [MDP slides](https://athitsos.utasites.cloud/courses/cse4309_fall2021/lectures/18_mdp.pdf).
* For terminal states, your model should not allow any action to be performed once you reach those states. For those states, you can just hardcode that their utility is equal to their reward.
* For blocked states, your code should capture the fact (by implementing the appropriate transition model) that they cannot be reached from any other state. You should hardcode the utility values of blocked states to be 0.

**Output**

At the end of your program, you should print out the utility values for all states, as well as the optimal policy.

The output should follow this format:

utilities:

%6.3f %6.3f...

...

policy:

%6s %6s...

...

More specifically:

* In your utilities printout, each row corresponds to a row in the environment, and you use the %6.3f format specifier (or equivalents, depending on the programming language) for each utility value. For blocked states, just print a utility of 0.
* For the policy printout, again each row corresponds to a row in the environment, and you use the %6s format specifier (or equivalents, depending on the programming language) for each action. To encode each action (or special state) use these characers:
  + For the four actions, use characters '<' for left, '>' for right, '^' for up, and 'v' for down.
  + For blocked states, use character 'x'.
  + For terminal states, use character 'o'.

Do NOT print out this output after each iteration. You should only print out this output after the final iteration. As an example, this is the output of my implementation, for <environment\_file>="environment2.txt", <non\_terminal\_reward>=-0.03, <gamma>=0.98, and <K>=20:

utilities:

0.781 0.846 0.906 1.000

0.724 0.000 0.646 -1.000

0.662 0.608 0.570 0.354

policy:

> > > o

^ x ^ o

^ < ^ <