For this project you will be implementing an agent to play a simple text-based adventure game. The agent is required to move around a rectangular environment, collecting tools and avoiding (or removing) obstacles along the way. The obstacles and tools within the environment are represented as follows:

Obstacles		**Tools**	
`T`	tree	`a`	axe
`-`	door	`k`	key
`~`	water	`o`	stepping stone
`*`	wall	`g`	gold

The agent will be represented by one of the characters `^`, `v`, `<` or `>`, depending on which direction it is pointing. The agent is capable of the following instructions:

`L`	turn left
`R`	turn right
`F`	(try to) move forward
`C`	(try to) chop down a tree, using an axe
`U`	(try to) unlock a door, using a key

When it executes an `L` or `R` instruction, the agent remains in the same location and only its direction changes. When it executes an `F` instruction, the agent attempts to move a single step in whichever direction it is pointing. The `F` instruction will fail (have no effect) if there is a wall, tree or door directly in front of the agent.

When the agent moves to a location occupied by a tool, it automatically picks up the tool. The agent may use a `C` or `U` instruction to remove an obstacle immediately in front of it, if it is carrying the appropriate tool. A tree may be removed with a `C` (chop) instruction, if an axe is held. A door may be removed with a `U` (unlock) instruction, if a key is held.

If the agent moves forward into the water it will drown **unless** it is holding a stepping stone, in which case the stone will automatically be placed in the water and the agent can step onto it safely. When the agent steps away, the stone will appear as an upper-case `O`. The agent can walk again on that stone, but the stone will stay where it is and can never be picked up again.

If the agent attempts to move off the edge of the environment, it dies.

To win the game, the agent must pick up the gold and then return to its initial location.

Running as a Single Process

Copy the archive `src.zip` into your own filespace and unzip it. Then type

```
cd src
javac *.java
java Stepping -i s0.in
```

You should then see something like this:

```
~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~
~~     T     T   k ~~
~~   ~~*     *~~   ~~
~~*-*     v     *-*~~
~~  **         **  ~~
~~ g **   o   ** a ~~
~~     *     *     ~~
~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~

Enter Action(s):
```

This allows you to play the role of the agent by typing commands at the keyboard (followed by <Enter>). Note:

- a key can be used to open any door; once a door is opened, it has effectively been removed from the environment and can never be "closed" again.
- an axe or key can be used multiple times, but each stone can be placed in the water only once.
- `C` or `U` instructions will fail (have no effect) if the appropriate tool is not held, or if the location immediately in front of the agent does not contain an appropriate obstacle.

Running in Network Mode

Follow these instructions to see how the game runs in network mode:

1. open two windows, and `cd` to the `src` directory in both of them.
2. choose a port number between 1025 and 65535 - let's suppose you choose 31415.
3. type this in one window:
4. `java Stepping -p 31415 -i s0.in`
5. type this in the other window:
6. `java Agent -p 31415`

In network mode, the agent runs as a separate process and communicates with the game engine through a TCPIP socket. Notice that the agent cannot see the whole environment, but only a 5-by-5 "window" around its current location, appropriately rotated. From the agent's point of view, locations off the edge of the environment appear as a dot.

We have also provided a C version of the agent, which you can run by typing

```
make
./agent -p 31415
```

Writing an Agent

At each time step, the environment will send a series of 24 characters to the agent, constituting a scan of the 5-by-5 window it is currently seeing; the agent must send back a single character to indicate the action it has chosen.

You are free to write the agent in any language you choose. If you are writing in Java, your main file should be called `Agent.java` (you are free to use the supplied file `Agent.java` as a starting point). If you are writing in C, you are free to use the files `agent.c`, `pipe.c` and `pipe.h` as a starting point. In other languages, you will have to write the socket code for yourself. You must include a `Makefile` with your submission, producing an executable called `agent`.

You may assume that the specified environment is no larger than 80 by 80, but the agent can begin anywhere inside it.

Additional examples of input and output files will be provided in the directory `hw3/sample`.