CSC322: Homework 1

This is your C lab. It consists of programming a small text-based game purely in C. The game was designed for OOP, which C does not support naturally. Thus this lab will simultaneously teach you the specifics of the C language, and the difference between structured programming and OOP.

Important: if you have already completed this project in another language, I recommend that you do not try translating your existing code into C. It will be a lot more trouble than starting from scratch.

You can find a complete description of the project in this pdf document. You can also download and execute the Java bytecode of the completed game to better get a grasp of it.

The description above says you should use XML as input. To make the project a little easier, I am changing a few things. First, there will be no names and descriptions for anything in the game, only numbers (as in room number 0); change the game output accordingly. In addition, I am changing the input definition to the following:

When the game starts, it takes from **stdin** (so no files are opened/closed/used in the project code) input in the following format:

n (integer, the number of rooms, 1 < n < 100)
The next n lines are the following:
state north south east west (five integers, signifying the cleanliness of the room and the numbers of the respective neighbors. State can be 0 for clean, 1 for half-dirty, 2 for dirty. -1 signifies no neighbor. Numbering of the rooms starts at 0)
The next line is:
m (integer, the number of creatures, 1 < m < 100)
The next m lines are:
creatureType location (two integers; creatureType is 0 for the PC, 1 for an animal, 2 for an NPC. Location is the number of the room the creature is in. Numbering starts at 0)

The input will be well-formed according to the above description and the game specification (i.e., you should not worry about checking for errors in the input such as putting an animal in a dirty room from the start and the like).

After the above is read from stdin, the game should continue reading game commands from stdin.

Make sure you use as few globals as possible. You must not use global arrays or global structs; you are allowed to use up to three global pointers and a global int for the respect value.

Here is an example input block:

```
3
0 -1 1 -1 2
2 0 -1 -1 -1
2 -1 -1 0 -1
4
1 0
0 2
2 2
2 1
look
clean
2:clean
east
look
0:clean
```

```
look
exit
```

To that input corresponds the following output:

```
Room 2, dirty, neighbors 0 to the east, contains:
PC
human 2

2 grumbles. Respect is now 39

2 grumbles a lot. Respect is now 36
2 leaves towards the east.

You leave towards the east.

Room 0, half-dirty, neighbors 1 to the south 2 to the west, contains:
animal 0
PC
human 2

0 licks your face a lot. Respect is now 39
2 grumbles. Respect is now 38
2 leaves towards the south

Room 0, clean, neighbors 1 to the south 2 to the west, contains:
animal 0
PC

Goodbye!
```

You should submit your source file(s) to the Angel dropbox. Your code should compile with the gcc on rho. If it requires any command-line parameters passed to gcc when compiling, specify them in the comment box in Angel when you make your submission.

Do not underestimate the project. It is sufficiently large to require all the time you have. Start early in order to finish on time.
