Building Z

Description

In the not-too-distant future a comet streaks across the earth's sky. Another near miss. Tensions ease. But in its wake will be a tail of disaster that no one will anticipate. You see, this comet passed through a region of space that was long before rendered lifeless by alien bio-wars.

The disease that will be shed on the earth's occupants is nothing like the world has ever seen. Sure, we will have been desensitized by countless zombie movies and a TV series that will go on 3 times longer than it should have. Perhaps it will be this desensitization that will lead to our undoing. The alien plague will cause humans to lose their minds. They will become predators that prey on the uninfected.

There is a military complex known as the box. It is located in an undisclosed location in Nebraska. The military will develop an antidote for the alien bioweaponry, but the complex will quickly become infected before the antidote can be dispensed. There is one scientist left who is uninfected. That scientist injected herself with the serum to prevent it from being destroyed, the other brave women and men who discover the cure will become the creatures who savagely attack anyone who attempts to complete their work. The surviving scientist suffered an injury and is incapacitated. The infected do not attack the scientist for some unknown reason.

Because of your mad skills as a C++ programmer, you will be hired to write a simulator to train the special operator who will attempt to penetrate the compound and liberate the scientist.

Intel

The scientist is located in the center of the compound's main building; called "Building Z." Getting into the compound is simple but finding one's way through a building occupied by maniacs of not.

There are 3 types of infected individuals. The first are referred to as "**Loungers**". They tend to be sedentary but occasionally move to an adjacent room. They are very, very strong and aggressive when they encounter a person who has not been infected. Our special operator will not survive an encounter with any of these. There are known to be 6 Loungers in building Z. The only move to a random adjacent room 20% of the time.

The second type of assailants are known as "Hypers". They are constantly on the move, but their movement is unpredictable (random). There are 4 Hypers in the building and they are stronger and more violent than the Loungers.

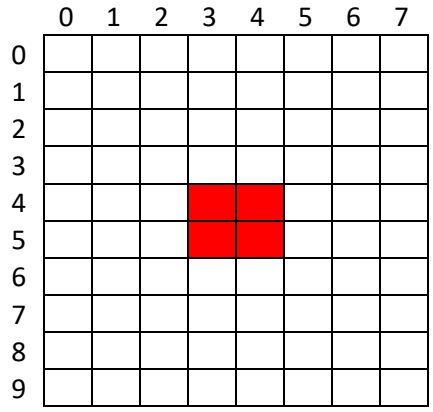
The Loungers and the hypers have a diminished sense of sight, hearing, and smell. They must be in the same room as an uninfected person to detect and attack them. The final type of infected people are known as "Aggressors." The do not suffer from the same reduced senses as the others. They are constantly on the move like the Hypers but they can sense a noninfected person from two rooms away. Fortunately, there are only 2 of these aggressors in the building.

Activity

Create a simulator program that our special operator will train with. The program has these requirements:

- Building Z has 80 rooms arranged in an 8x10 grid.
- The "spec-op" always starts the rescue in room (0,0) which has an exterior door. The scientist is located in one of the 4 centrally located rooms.
- Moves can only be made to directly adjacent rooms. The attackers and the spec-op can only move up, down, left, and right.

- The following number and type of infected people are randomly located throughout the building:
  - 6 Loungers
    - For each move our spec-op makes, the Loungers have a 20% chance of moving to a random adjacent room.
  - 4 Hypers
    - With every move the spec-op makes, the hypers move as well, but in a random direction.
  - 2 Aggressors
    - The aggressors move like the Hypers. They will sense the spec-op at a distance of two rooms or less and will pursue.
- The antidote is always located in one of the center rooms in the building. These rooms are shown in red in the diagram below. With the start of each play, the antidote will be placed randomly in one of these rooms.



- More than one infected people can be in a room at the same time. They can be in the room with the antidote as well.
- Wall-penetrating radar and infrared satellite imagery will provide you with a steady stream of data. This means that all the people are visible at all times, as is the scientist.
- There are 6 actions that our special operator can take: move west, move east, move north, move south, pause (don't move one turn), and carry (to pick up the antidote), [N, S, W, E, P, C]. Of course, C (carry) uses a turn while our special operator picks up the scientist. If C is entered when the spec-op is not in the room with the antidote, or the spec-op is already carrying it, then the action will be the same as P (pause).
- Loungers will be represented as an "L". A Hyper is represented by an "H", and an Aggressor by an "A", the scientist by an "S". and the special operator by an “O”.

Design

- You must create a building class.
  - This class will be the controller for the entire simulation.
  - It will use aggregation to hold the 12 Infected objects in a collection of Infected pointers. Use a std::vector for this collection.
  - It will also hold the special operator and the Scientist.
  - The only way the scientist can move is if he/she is carried by the special operator.
  - Make sure the building class has a destructor to delete all the objects created with the new keyword when it goes out of scope.
  - The building will have a public function called, "move".
  - The building will also have a friend function that overrides the stream insertion operator (<<). This function will insert into an outstream the layout of the building with the position of all of the Person objects in it.

- Create a struct called Position which has two public data members, x and y. This will hold the cartesian coordinates of each Person. This struct must have a constructor that takes in the initial values of x and y and a copy constructor that takes a position as an argument.
- Create a person class that will be the base class for all the infected, the spec-op, and the scientist.
    - This class will have a protected Position data member which represents the Person's current position.
    - It will also hold the type of person in the form of a protected char.
    - This class also has a constructor that takes a Position and type as its only arguments. This will receive the initial position of the of the person and the type of person. Override the equality and inequality operators, == and !=. These compare the position of two Person objects.
    - This class also has a pure virtual function called move which takes the move as its argument which has a default value of 'A' for auto.
- Create an Infected class that derives the person class.
    - This class will have a pure virtual function called move which has a move as its parameter which defaults to 'A' for auto.
- Create the three Infected classes, Loungers, Hyper, and Aggressor which derive the Infected class.
    - These three specialized versions of the Infected class will override the *move* function in Infected.
    - Each specialized overridden *move* function ignores its position parameter and calculates its own move or decides not to move at all as in the case of the Loungers. These moves must be legal moves.
    - Their constructors take as a parameter their initial position.
    - The constructor for the Aggressor class also takes a second parameter which is the pointer to the spec-op object. This way the Aggressors know where the spec-op is at all times and can hunt her/him down.
- Create a Uninfected class that inherits Person. This is the class that represents our unconscious scientist and the spec-op.
    - This class overrides the pure virtual *move* function in its inherited Person class. The move function must accept only a valid Position to which to move. It should throw a runtime exception if the move is invalid.
- Write a program called building_z.cpp which handles all user input and output as well as turn taking and the player's move making.
    - When the main program needs to output the state of the Building object (we'll assume it's called "buildingZ" for this explanation) it will be inserted into the cout stream like so, std::cout << buildingZ << std::endl.
    - The program should prompt the user for a valid move and re-prompt if an invalid move is entered. Valid moves are listed above but restricted by the spec-op's position in the castle. In other words, if our spec-op is at (0,0) she/he can only move east or south.
    - The spec-op saves the scientist if both of their positions are at 0,0.
- The headers for all classes will be included with this project but might be incomplete in terms of inheritance and the virtuality of functions.
    - You may have to declare the inheritance and make functions virtual or pure virtual as needed.
    - All objects will be created on the heap so the destructor in the object or program that created an object is responsible to delete it.
    - You must declare include-guards for all the classes and the struct.
    - You must write the classes definition in a separate cpp file.

Design Notes

- Stay true to the requirements of this project. This is meant to gain a better understanding of inheritance and polymorphism.
- You can create private helper (utility) functions in any of your classes but do not add any public functions.
- Create all objects, with the exception of container objects, on the heap. Use the new keyword for this but be sure delete them from the destructor of the object responsible for them.
- Always try to code for efficiency.
- The start of play should look like this:

```
Rescue mission in progress
---------------------------------
|   |   |   |L  |   |   |   |   |
| O |   |   |   |   |   |   |   |
---------------------------------
|   |L  |   |   |   |LH | H |   |   |
|   |   |   |   |   |   |   |   |   |
---------------------------------
|   |   |   |   |L  |   |   |   |   |
|   |   |   |   |   |   |   |   |   |
---------------------------------
|   |L  |   |   |   |   |  A|   |   |
|   |   |   |   |   |   |   |   |   |
---------------------------------
|   |   |   | H |   |   |   |   |   |
|   |   |   |   | S |   |   |   |   |
---------------------------------
|   |   |   |   |  A|   |   |   |   |
|   |   |   |   |   |   |   |   |   |
---------------------------------
|   |   |   |L  |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |
---------------------------------
|   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |
---------------------------------
|   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |
---------------------------------
|   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |
---------------------------------

Your move [N,S,W,E,P,C]:
```

- **Note, if there are more than one of any person in a room, only display one letter for each type this way you can fix the position of each type in the output.**

Requirements

- **Do not change the user interface or the behavior outlined in these specifications.**
- **Add pre and post condition comments to function prototypes in each header file.**
- **Create a separate definition file for each header file.**

- **You can add private members to the header file of any class but do not change the public or protected sections unless you get permission.**

Reminder

You are responsible to do your own work. **This is not a team project**. **Do not show your code to anyone and do not look at, or copy, any code from any source (even the lecture notes or the book); create your own code.** Our lectures and the book contain all the information you need to complete this project. Any violation of the school's academic integrity policy or the policy of this class will result in a zero grade and an academic misconduct report filed with the school; no excuse will be accepted.

Your program must compile and run to receive any credit. If I cannot compile your program, you will receive a zero for your score. Treat this like any other exam, start right away and put effort into it.

Rubric

This project/exam is worth 200 points. The points will be distributed according to the chart below. If the program does not compile or is missing, then no points will be awarded.

Partial credit will be awarded for the features of the program that do work if the program compiles and runs.

Program 200
Requirements 100
Correctness 80
Code Quality 20
Total 200

Note: code quality refers to following good coding practices. Indentation, spacing, proper naming of identifiers, and modularization make up this part of the grade.

What to Submit

- All your header and source code files

Hint: the sooner you get started the more you will enjoy this project and the better you will do. No late projects will be accepted. I will offer help and hints if you start early but please do not ask me to look at your code.