Program #1: Vacuum Cleaning Agent

Due: Friday, Oct. 11

Your assignment is to write an intelligent agent for a variation of the vacuum cleaning world described in the textbook. Like the robot in the book, your robot is capable of sucking up dirt, moving forward and turning right. It is also able to sense if it is over any dirt. However, we will use a more complex environment, in that there are obstacles that may block the agent. Fortunately, the robot has a sensor that can detect if an obstacle is directly in front of it. Another sensor can detect if the agent has accidentally bumped into an obstacle.

The robot's task is to make sure the entire room is clean. When this task is successfully completed, it should return to its starting position and turn itself off (this can be done with the robot facing any direction it chooses). The agent should be intelligent in that it can perform this task efficiently in new situations. In other words, the robot does not know the location of the obstacles or dirt ahead of time, but must make sure that the room is free of dirt using as little power as possible. This means it should try to minimize its movements and only turn on its suction device when necessary. The robot should also avoid bumping into things, because this can damage valuable personal articles and/or the robot itself.

The figure below shows an example initial state for the room. The robot is indicated by the "A", dirt is indicated by "*", and obstacles are indicated by an "X". The agent starts in the upper left corner of the room in square (1,1) and is facing south. Note that the room itself is 5x5 and is completely surrounded by obstacles (walls). In this case, there are also obstacles (furniture) in the upper right and lower left corners of the room.

```
  0  1  2  3  4  5  6
 +--+--+--+--+--+--+--+
0|X |X |X |X |X |X |X |
 +--+--+--+--+--+--+--+
1|X |A |  |  |  |X |X |
 +--+--+--+--+--+--+--+
2|X |  |* |  |* |X |X |
 +--+--+--+--+--+--+--+
3|X |  |  |  |  |X |X |
 +--+--+--+--+--+--+--+
4|X |X |  |  |  |  |X |
 +--+--+--+--+--+--+--+
5|X |X |X |  |* |  |X |
 +--+--+--+--+--+--+--+
6|X |X |X |X |X |X |X |
 +--+--+--+--+--+--+--+
Location: (1,1)  Facing: SOUTH
```

Although the robot must be able to cope with a number of different situations, we will put some constraints on the task to make it feasible. First of all, the agent will only be used in 5x5 rooms that are completely surrounded by walls. Second, the agent will always start in square (1,1) facing south. This is also the square the agent must return to when it completes its task. Finally, it will be able to reach any dirt in the room. Otherwise, dirt and obstacles may be in any open square.

In order to help you test your agent, I have written a Java Vacuum World simulation using the design we worked on in class. This source code can be downloaded from our course web page (http://www.cse.lehigh.edu/~heflin/courses/agents-2002/). Everything is provided for you, except you must write the **VacAgent** class yourself. This class should be in the **vacworld** package, be a subclass of **Agent**, and should implement the **see()** and **selectAction()** methods.

Before each move, the agent receives a **VacPercept** by way of its **see()** method. This percept contains information about whether the robot sees dirt below it (the **seeDirt()** method), sees an obstacle directly in front of it (the **seeObstacle()** method) and if it bumped into an obstacle on its last turn (the **feelBump()** method). For details, see the **VacPercept.java** file.

When the robot's **selectAction()** method is called, it must return one of four actions. Each type of action is a subclass of Action. These classes are:

- **SuckDirt** – This action will remove dirt from the square the robot is in
- **GoForward** – This will move the robot forward one square in the direction it is facing. If it is facing an obstacle then the robot will not move, but will feel a bump in its next percept.
- **TurnRight** – This will turn the robot 90 degrees to the right.
- **ShutOff** – This will cause the robot to power down. The robot should only execute this action when it has cleaned the room of all dirt and returned to its starting position.

After you have compiled your **VacAgent** file in the vacworld directory, you can start the simulator by typing "`java vacworld.VacuumWorld`" (assuming you are in the directory directly above `vacworld` and have "." in your Java `CLASSPATH` environment variable). The simulator outputs a map of the state of the world to the console. To advance to the next state, simply press RETURN. The percept received by the agent and the action that it selects will be printed, along with the resulting map. If this primitive-looking interface bothers you, feel free to modify the code with a more graphical interface, but make sure that your program also works with the original code, because that's what I'll be using to test it.

You will primarily be graded on your agent's ability to complete the task (that is, clean the room and return to the starting position). I will test it with multiple room and dirt configurations, one of which will be the one shown in this assignment. The degree of intelligence that your agent displays will also be factored into your grade. Agents that are able to complete all tasks with fewer moves, bumps, and use of suction will be graded higher than ones that needlessly waste energy. Note that you don't have to have an optimal solution to get an A on this assignment, but a near optimal solution would help. Finally, the elegance of your design will be considered in your grade as well. However, this design need not be based on an architecture described in the book; it could be something you developed yourself.

You must submit your assignment to me by the beginning of class on Friday, October 11. There are hardcopy and electronic components to your submission. Your electronic submission must consist of the source code (.java files) and compiled (.class) files for VacAgent and any other supporting classes you developed. I expect you to have a descriptive comment for each class and method, as well as comments to explain any complicated logic you might have. You may send this by e-mail to heflin@cse.lehigh.edu or you can submit it on a 3.5" floppy disk. You must also hand in a hard copy of your source code.