Homework 2: A Classy Karel

Due Date: Thursday, September 18, 2008.

There is a possible 45 points for this homework.

Problem 1. (10 pts.) Program a robot named Karel, starting at the origin, to locate a treasure (beeper) in a maze. Figure 1 shows one example of a legal maze. Your program should work with any closed maze.

![Sample Maze World](image)

Figure 1: Sample Maze World

Your score will be dependent on how sophisticated your “maze walker” behaves.

1. (6 pts.) Your robot can successfully find the treasure beeper in the example maze.

2. (10 pts.) If your robot can successfully locate the treasure in the maze, regardless of where the treasure is, and regardless of the maze’s configuration (as long as it’s legal).

3. (4 pts Extra Credit) In addition to successfully locating the treasure, the robot successfully returns to the starting position; the origin. The treasure beeper can be anywhere in the maze except on the origin (the starting position), hence one can drop a beeper at the origin to “find” one’s starting position after finding the treasure.
Problem 2. (25 pts.) Teach (i.e. program) Karel to play fetch. Program Karel, starting from the origin (1,1) facing east, to jump over hurdles, pick up the beeper just beyond the hurdles, return to the origin and place the beeper there (i.e. at you feet). Hint: One can recognize the origin by dropping a beeper there prior to fetching the “thrown” beeper.

Your score will be dependent on how sophisticated your “fetcher” behaves.

1. (10 pts.) If your robot can only work in a world such as the one illustrated in figure 2 that has a fixed number of hurdles. Each hurdle is a fixed height and width, and each hurdle is a fixed number of avenues apart.

2. (15 pts.) If your robot can work in a slightly more complicated world with a fixed number of hurdles. Each hurdle can now be a different height and/or width, but each hurdle is still a fixed number of avenues apart.

3. (20 pts.) If your robot can work in a still slightly more complicated world. Not only can each hurdle be a different height and/or width, but you don’t know how many hurdles there are. Furthermore, the hurdles can be separated by a varying number of avenues. Figure 3 is one example of what this world could look like.

4. (25 pts.) If your robot can work in the most complicated of fetch worlds. Not only can there be a variable number of hurdles, each a different height and/or width with a varying inter-hurdle distance, but the beeper to fetch can be anywhere “on the ground” instead of two avenues east of the final hurdle. (The beeper to fetch will never be on the sides or on top of a hurdle.)
**Problem 3.** (10 pts.) Karel the robot meets Carol the robot on a corner. Have Karel determine whether it has exactly the same number of beepers in its beeper-bag that Carol has. Each robot should finish with the same number of beepers it starts with. Karel should end facing north if it has the same number of beepers as Carol, and should end facing south otherwise.

**Problem 4.** (10 pts. Extra Credit) Program a robot name Karel to arrange vertical piles of beepers into ascending (or descending) order. Each avenue, starting at the origin, will contain a vertical pile of one or more beepers. The first empty avenue will mark the end of the piles that need to be sorted. Figure 4 illustrates one of the many possible initial situations and corresponding final situation.

**IMPORTANT:** For each problem you solve, create one or more classes which inherit either directly from the `Robot` class, or better, your own `myBetterRobot` class. It is highly recommended that for each problem you create a class hierarchy that solves successively more complicated versions of the task. In addition to correctness (and documentation), you will be graded on your class decomposition/design.

To submit your programs you need to:

1. print out a hardcopy to physically hand in on the due date.
2. email the code to your instructor for testing (mikeyg@cs.xu.edu or ejohnson@cs.xu.edu)

There will be a 5 point penalty for not doing both submission tasks.
Figure 4: Initial and Ending Configurations