Problem 1

In this problem you will complete a program and then answer two questions dealing with passing parameters.

Complete a function `modifyGrid` that places blocks on all outer edges of a 5×5 grid and things on all inner intersections except where the mover is (in row 2, column 2). When you have finished, change the parameter from a reference parameter to a value parameter by deleting the `&`. What is the consequence? Next, change `ioGrid` to a `const` reference parameter. What is the consequence?

```
#include "/home/hydra/COURSES/cs203/include/grid"

/*****************************************************
*                                                    *
* Pre:  ioGrid is a 5x5 grid with the mover in row 2,    *
*       column 2                                     *
* Post: ioGrid has the entire outer edge blocked with #  *
*       and there is a cookie on the other rows and      *
```

```
*       columns.                                                *
*                                                               *
*****************************************************************/
void modifyGrid(grid & ioGrid) {
    // Complete this
}

int main() {
    grid tarpit(5, 5, 2, 2, north);

    modifyGrid(tarpit);
    tarpit.display();

    return 0;
}
```

The output from your program should look like this:

```
The grid:
```

```
# # # # #
# O O O #
# O ^ O #
# O O O #
# # # # #
```

Problem 2

Write a program that constructs a hurdle course, as shown in the first grid at the end of this problem, and instructs the runner to jump the hurdles. The runner must touch the ground between each hurdle. Show the grid before and after the course is run. First retype the following test driver and then implement the functions `initializeGrid` and `jumpHurdle`.

```
#include "/home/hydra/COURSES/cs203/include/grid"

// post: the mover has jumped one hurdle
```

```
void jumpHurdle(grid & ioGrid)
{
    // Complete this
}

// post: Return a 4 x 22 grid ready to run a 4 x hurdle
//       race.
grid initializedGrid()
{
    // Complete this
}

// Test drive initializedGrid and jumpHurdle
int main()
{
    grid jumper = initializedGrid();
```

```
    jumper.display();
    jumpHurdle(jumper);
    jumpHurdle(jumper);
    jumpHurdle(jumper);
    jumpHurdle(jumper);
    jumpHurdle(jumper);
    jumper.display();

    return 0;
}
```

Your output should look like this.

```
The grid:
. . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . .
> . . # . . . # . . . # . . . # . . . # . .
. . . . . . . . . . . . . . . . . . . .
```

```
The grid:
. . . . . . . . . . . . . . . . . . . .
. .       .       .       .       .     .
      #       #       #       #       # > .
. . . . . . . . . . . . . . . . . . . . .
```