CS2303 Systems Programming Concepts A08

Program 2 36 Points
Due: September 15, 2008 at 11:59 p.m.
Simulated Robot Motion

This assignment emphasizes the use of ***arrays*** in C and provides practice with manipulating subscripts and abstracting with 3-dimensional arrays. You may use ***structs*** in this assignment but it is possible to complete this program successfully only using ***arrays.*** Moreover, since this assignment is being done early in the course, students can keep the building array and the current robot location as global variables. However, this does **NOT** preclude the expectation that the control structure of this program will consist of many small functions and routines where appropriate parameters are passed.

Note – While the program assignment is given with fixed sizes for floor plans, the expectation is that students will use ***#define*** statements such that the program can easily be adapted to variable sizes by simply changing a few ***#define*** statements.

Your task is to simulate the motion of a sequential series of robots that move on three floors of a four-floor storage building getting and putting objects into storage bins. The floor plans for the First, Second and Fourth Floors (The Third Floor is left empty for future expansion!) are given at the end of this assignment. The storage bins are numbered on the floor plans. Storage bins on the First Floor can **ONLY** be accessed by the robot from the row above a specific bin. Storage bins on the Second Floor can **ONLY** be accessed by the robot from the column to the left of the bin. Storage bins on the Fourth Floor can be accessed from **any** square adjacent to the bin that a robot can traverse. Once a robot arrives at the spot where it can access the target bin, the robot remains at that spot **×** time units to simulate accessing the bin where **×** equals the floor number. For example, if the robot's target bin is 2306, after the robot reaches the accessible spot, it remains at that spot two additional time units because this bin is on the Second Floor.

Robot motion on all three floors is quite restrictive. To reduce potential robot collisions, the floor plans indicate one-way and two-way paths that a robot can travel on each floor. Robots can ONLY move on the marked paths.

This program provides a simulation of the time it takes all the robots to complete their assigned tasks. Time will initially be set to 0. It takes **1 time unit** for the robot to move one square on a given floor. It takes **1 time unit** to move from a square to a stairs square or from a stairs square to a square. It takes **2 time units** to use the stairs between any two floors. For example to move from the 1st floor stairs square to the 2nd floor stairs square takes 2 time units while it takes 4 time units to move from the 2nd floor stairs square to the 4th floor stairs square.

Assignment Input

The program will read in 'scripted' information for the robot simulations from an ASCII file. To keep it simple, do **not** use C file I/O commands. The file should be accessed by using UNIX to redirect the input from a file. This will allow you to test the program either using a file or using terminal input.

{Think of the scripted input as consisting of subsequent lines of input data}

The program begins by reading in the number of robots that will be sequentially simulated. This determines how many more lines of input exist in the script file. For each robot, the program will read in a line of information of the form:

```
xpos ypos zpos bins bin-list
```

where

`xpos ypos zpos` indicate the start location for the robot. `zpos` indicates the floor and `xpos ypos` indicate the location on that floor. {Assume the upper left corner position on a floor is (0, 0).}

`bins` is the number of bins the robot will access.

`bin-list` is the robot's list of bins to access. Assume the bin-list is `sorted` from low to high by bin number.

For example, given the input line:

```
4 7 2 5 1106 1407 2305 2406 4402
```

the robot will start from an initial location on the Second Floor in location (4, 7) which is just to the left of bin 2404 (see the Second Floor plan). This robot's assignment is to access the five bins: 1106, 1407, 2305, 2406 and 4402.

Robot Motion Planning

Beginning from its initial location, each robot will access its assigned bins in increasing numerical order by obeying the travel rules specified on the floor plans. Your program should use the shortest path for the robot to access its bin-list in numerically increasing order regardless of the robot's initial position.

Note - robots cannot travel 'through' bins. To go to a different floor, a robot must go to and use the stairs.

Thus, a robot's life begins at its initial location and moves to access all its assigned bins using the stairs when switching floors. When the robot has accessed the last bin on its bin list, it finishes by returning to the stairs on the First Floor. At that point, the i^{th} robot is **instantly eliminated** from the simulation. One time unit later, the $(i+1)^{th}$ robot is placed at its initial location and the simulation continues. When the last robot completes its task and arrives at the First Floor stairs, the simulation ends.

Assignment Output

For each robot in the simulation, the program should print out a log of the robot's significant events that includes the following information, but not necessarily in this format:

robot `i` begins at location `(x,y)` on floor `z` at time `t`.

robot i arrives at bin b at time t.
robot i is going from floor z1 arriving at floor z2 at time t.
robot i left the simulation at time t.

Additionally, interleaved into the output is a **periodic** output showing the robot's position on a two-dimensional map for the robot's current floor. The correct floor map should be printed out every 25 time units (i.e., for time = 0, 25, 50, 100, 125, …)

After the simulation ends, print out a summary of each robot's time in the simulation.
For example:

```
Robot       Start Time  Finish Time
  1             0           65
  2            66          134
  …
```

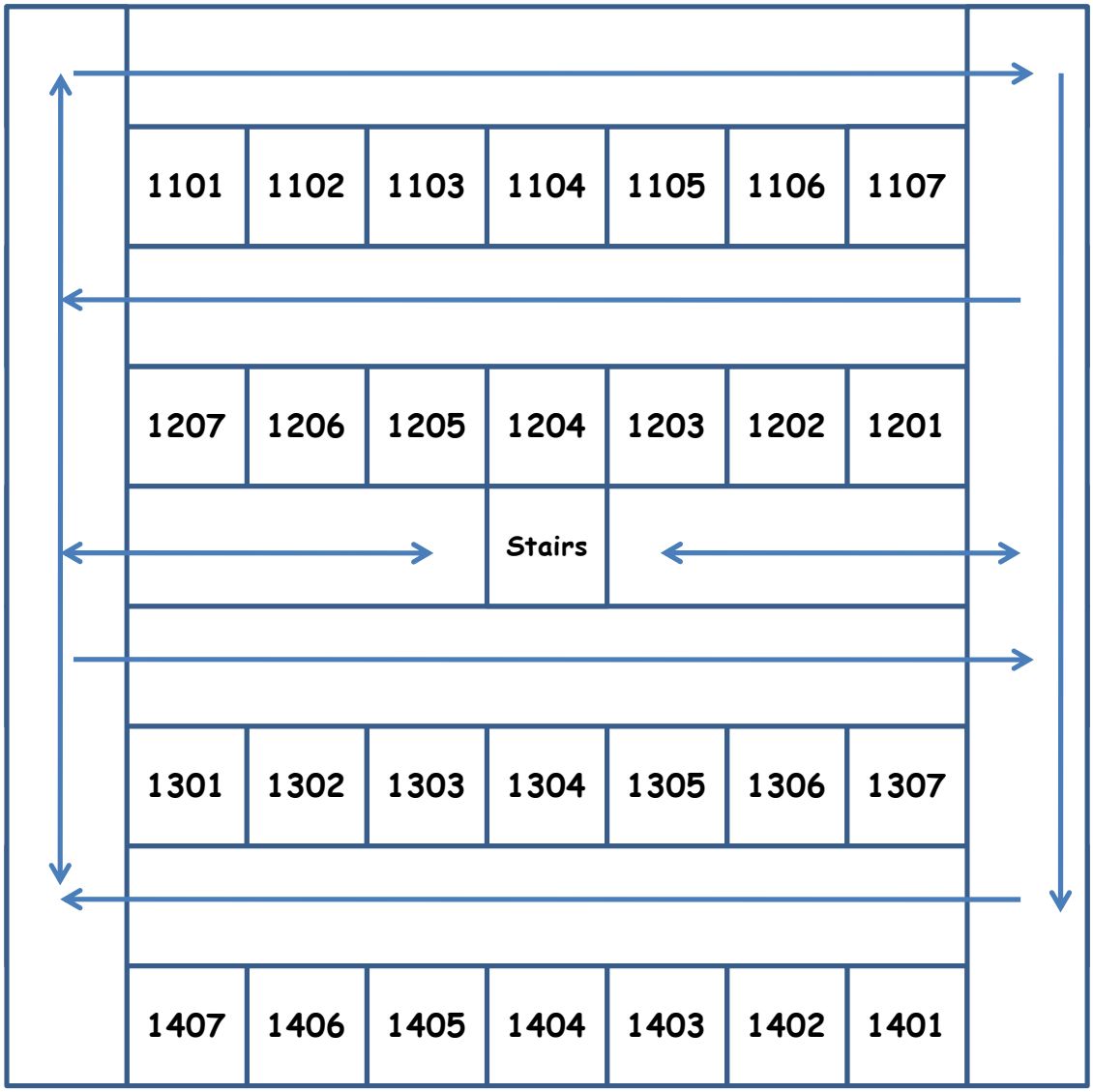
Program Development Hints

Remember this program will be graded taking into account good organization and the use of multiple functions and routines. It is highly recommended that students utilize a debugger and/or include a conditional define debug option that can easily be turned on and off. While debugging your program, you should print out other significant events to assist you in tracking the robot.

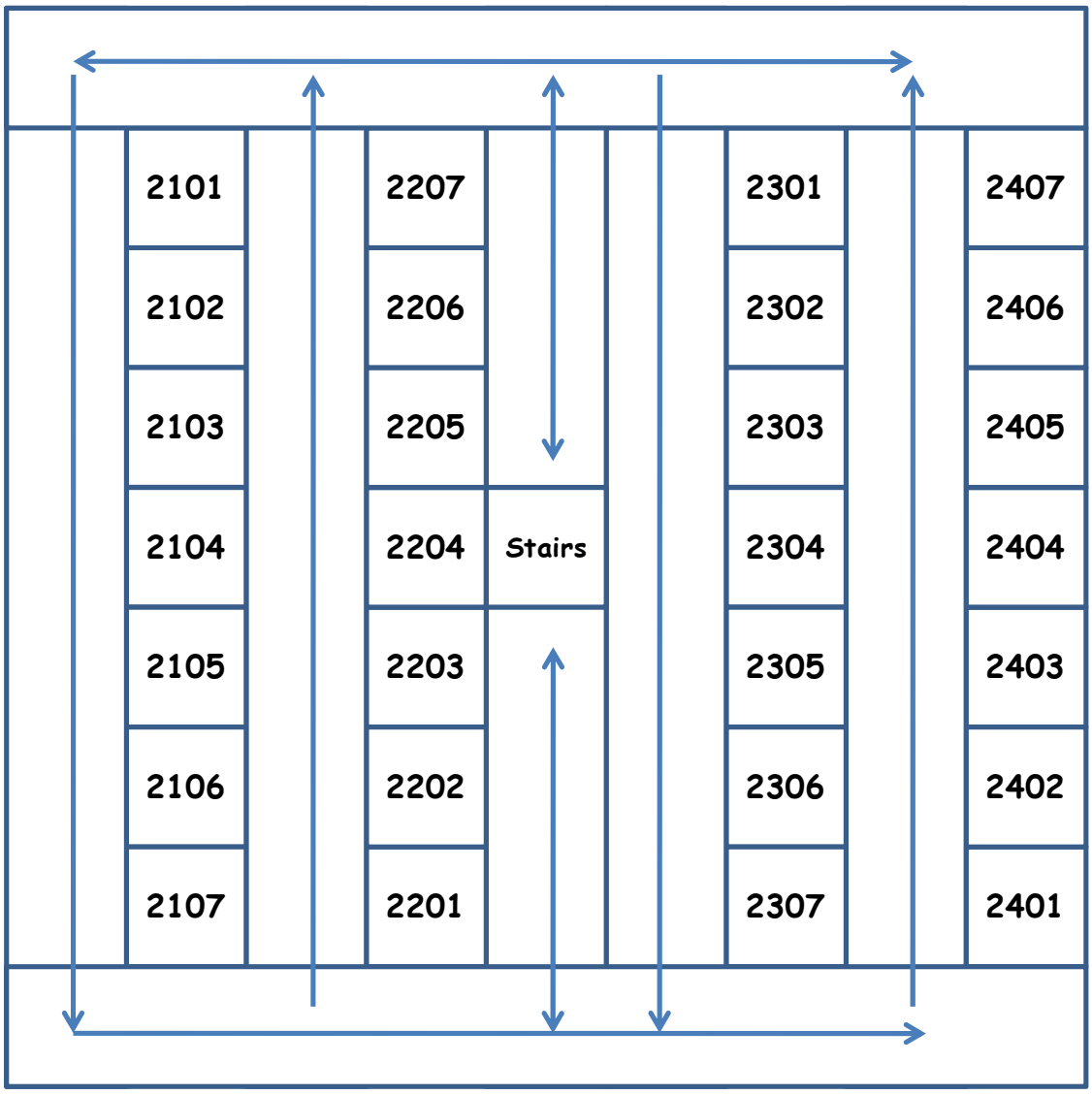
What to turn in for Program 2

Turn in your assignment using the ***turnin*** program on the CCC machines. You should turn in a tarred file that includes your source code, a make file, a README file and a sample output file. The make file should include the ability to cleanup leftover output and intermediate files between compile and execution cycles. The README file should provide any information to help the TA or SA test your program and evaluate your output files for grading purposes. If your program does not work successfully for a given floor, please indicate this in your README file.

First Floor


1101 1102 1103 1104 1105 1106 1107
1207 1206 1205 1204 1203 1202 1201
Stairs
1301 1302 1303 1304 1305 1306 1307
1407 1406 1405 1404 1403 1402 1401

Second Floor



Fourth Floor