ECE 3090 Fall Semester, 2012

PROJECT 3 – ROBOT SEARCH AND RESCUE

Assigned: Sept 28, 2012 Due: Oct 19, 2012 – 1:05pm

You were so successful at programming your robot explorer that NASA has called on you to program their next generation of rovers. You accept the challenge graciously and your robotic explorer is sent to some distant 2-dimensional planet. Unfortunately, your robot has been assigned to explore the most boring planet in the known universe. Having a perfectly rectangular surface, the planet holds no surprises for a would-be explorer. However, your robot has recently received a distress call from an astronaut who has crash-landed somewhere on the surface of the planet. Your goal is to program your robot to utilize its available information to navigate from its current position to the astronaut's crash site. You must do this by applying a search method to find a set of grid points that enable your robot to traverse from the start to the goal position.

Input

Obviously, for your search algorithm to function correctly, it needs to load the map of the environment into memory. The world map will be stored in the file "input.dat" and is defined by the following parameters:

MAP width height
: Defines the width and height of the world where (0,0) represents the top, left corner of the map

ORIGIN x_position y_position theta
: Defines the starting (x,y, θ) position of the robot where θ is the heading direction of N, S, E, W

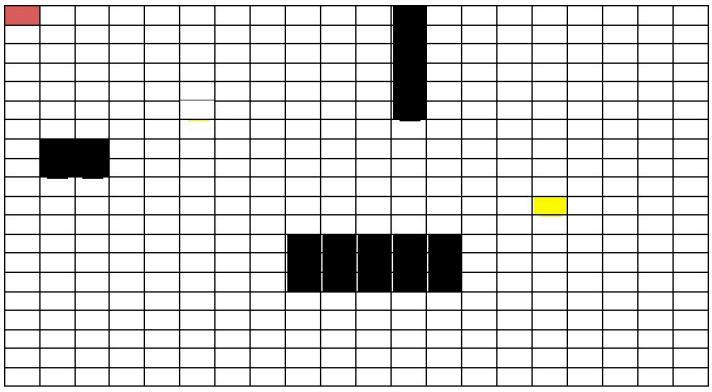
GOAL x_position y_position
: Defines the (x,y) location of the goal position of the astronaut crash site

OBSTACLE x_position y_position width height
: Defines the width and height of an obstacle, starting at (x,y). These are grid locations that the robot cannot navigate through.

Here is an example "input.dat" with a corresponding visual map (for illustration).

.

```
MAP 20 20
ORIGIN 0 0 E
GOAL 15 10
OBSTACLE 8 12 5 3
OBSTACLE 1 7 2 2
OBSTACLE 11 0 1 6
```



Output
Given the map environment, your program must output to "output.dat" - 1) the sequence of *x, y* grid points that transition the robot from the starting position to the goal position, 2) the total number of turns (i.e. direction changes) from start to goal, and 3) the total number of grid cells traversed. Note - the robot can only navigate in the North, South, West, and East direction and cannot navigate beyond the boundaries of the world.

Grading Policy: Although there are no requirements on design of the algorithm, you must use classes for this program. The code for each class definition must be stored in a separate file (i.e. Class1.cpp, Class1.h). The project will be graded based on the following criteria.

Optimal credit (20 points for each case): Program outputs an optimal solution (if one exists) for the robot. In this case, the program must run within 1.5 minutes and provide the shortest trajectory (in length, then in number of turns) from start to finish based on the available navigation directions. Deductions will be taken for crossing through an obstacle or a wall.

Suboptimal credit (17 points for each case): Program provides a solution (not necessarily the shortest trajectory) and terminates within 1.5 minutes. Deductions will be taken for crossing through an obstacle or a wall.

Partial credit (14 points for each case): The robot attempts to provide a solution but does not terminate within 1.5 minutes. In this case, your program must output a sequence of sequential grid locations that start from the initial position (but may not necessarily end at the goal position). Deductions will be taken for crossing through an obstacle or a wall.

***Additional Deductions**:*

- Program crosses through an obstacle or a wall (3 pts for each occurrence)
- Program will not compile (60 pts)
- Program crashes or does not terminate (20 pts)
- Program does not contain at least one class file (20 pts)
- Program does not comply with requirements or good design constraints (e.g. non-working makefile, no zip file, inadequate comments, use of global variables, etc.) (2 to 10 pts)

Submission: Your project code is to be submitted via the ECE3090 T-Square site under Assignments-Project3 on or before the due date. All relevant files (including a makefile to compile your code) should be zipped up and named by firstName_lastName_project3.zip. We will test your code on the Jinx cluster so make sure your program correctly compiles and runs on that system. Information on the cluster is located at: http://support.cc.gatech.edu/facilities/instructional-labs/jinx-cluster.