










Due: Nov 16/21, 2012 – 1:05pm

	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5										
6										
7										
8										
9										

Output

Given any *input.dat* consisting of a start location and set of cans, Robby should learn the necessary actions required to navigate the environment while collecting empty soda cans. For example, if Robby has learned the following control strategy:

Situation					Action
North	South	East	West	Current Site	MoveNorth PickUpCan MoveRandom MoveWest
Empty	Empty	Empty	Empty	Empty	
Empty	Empty	Empty	Empty	Can	
Empty	Empty	Empty	Empty	Wall	
Empty	Empty	Empty	Can	Empty	PickUpCan
:	:	:	:	:	
Wall	Empty	Can	Wall	Empty	StayPut
:	:	:	:	:	
Wall	Wall	Wall	Wall	Wall	

Then, if Robby's current location corresponds to the situation:

<i>North</i>	<i>South</i>	<i>East</i>	<i>West</i>	<i>Current Site</i>
Wall	Empty	Can	Wall	Empty

the corresponding action would be: *PickUpCan*

Given the map environment, your program must output to "output.dat" - 1) the sequence of *x, y* grid points that Robby traverses, 2) the corresponding action at each of those grid points, 3) total number of actions, 4) total number of cans collected, and 5) the total score for the traversed sequence.

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 strategy based on collecting all of the possible cans in the minimum number of actions. Remember that your robot is bounded by a maximum number of actions. The program must run within 1.5 minutes.

Suboptimal credit (17 points for each case): Program outputs a sub-optimal strategy based on collecting all of the possible cans but not in the minimum number of actions. Remember that your robot is bounded by a maximum number of actions. The program must run within 1.5 minutes.

Partial credit (14 points for each case): Program outputs a sub-optimal strategy based on achieving a correct score (as discussed above) bounded by the maximum number of actions. If you believe your program will take longer than 1.5 minutes to run, you must indicate that via a message to the console.

Additional Deductions:

- Program will not compile (60 pts)
- Program crashes or does not terminate (20 pts)
- Program does not contain at least one separate 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-Project5 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_project5.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>.

Extra Credit: Submission of your working code by the 19th of November is worth 10 points.