COP 3530 Fall 2013

Data Structures and Algorithms

Assignment 4. Due: October 14, 11:55 p.m. via Sakai

In this problem you will solve the “Rat in a maze problem”, using Stacks and Queues (Lectures 12-14).

The main points we shall be covering are:

* Using Stacks and Queues in an application
* Re-enforcement of the usage and advantages of makefiles / make utility in UNIX/Linux
* Use of abstract data types in C++, and separate compilation
* Use of header files and libraries for Stacks and Queues

The “Rat in maze problem” consists of finding a path in a maze, with given starting and ending points. The maze is given as a 2D array, where an open path is a sequence of positions represented by ‘0’s. The rat(s) cannot move to walls which are represented by ‘1’s. For every position “taken” by the rat, the position changes from ‘0’ to ‘2’. The coordinates of the square at the top left corner are (0,0). The coordinates of the square at the bottom right corner are (12,14). For example:

(0,0)

000100000010000

000100100000000

000000011111111

000111001001000

000001100100100

110001100100000

011001100100000

001001100100000

011001000000000

001000111111111

001010000000000

001010001000000

000010010000000

(12,14)

You will implement a new class (in corresponding .cpp and .h files) called RatInMaze. This class stores the maze, and provides three methods:

**searchStack:** A rat that always attempts to move in a specific order: right, down, left,

up. This method should be implemented using a stack.

**searchQueue:** On a given square, the rat reproduces, and (up to) four rats take one step away from the current square, each rat in a different direction. Each of these rats will choose one direction: right, down, left and up, **in that order**. This method should be implemented using a queue.

**searchStackSmart:** A rat that first attempts to move in the general direction of the exit,

and moves away from the exit only when moving towards the exit has failed. For

example, if the exit is down and right from the rat, the order of the first two directions it

will attempt will be either “down, then right” or “right, then down”. This method should

also be implemented using stacks. **You may attempt the next move according to both direction and distance, but only direction is required. Therefore, your results on this search, ONLY this search, may differ from ours (see below).**

These methods find a path from (fromX,fromY) to (toX,toY), simulating the actions of a rat using the following different strategies:

bool searchStack(int fromX,int fromY,int toX,int toY) ,

bool searchQueue(int fromX,int fromY,int toX,int toY), and

bool searchStackSmart(int fromX,int fromY,int toX,int toY)

Your implementation of stacks, as well as queues, is to be based on Arraylist’s.

Files provided by instructor: (see the Announcements page)

* main.cpp. Creates the maze, and invokes the three rat search algorithms.
* Linear**l**ist.h, Arraylist.cpp, and Arraylist.h. Implementation of ArrayList.
* myException.cpp and myException.h.

Files you must develop

* stack.cpp and stack.h. Stack implementation, based on ArrayList’s.
* queue.cpp and queue.h. Queue implementation, based on ArrayList’s.
* RatInMaze.cpp and RatInMaze.h. Implementation of the maze, and three search algorithms.
* Makefile. Each .cpp file must compiled separately. No templates. Be sure to provide “clean” and ‘tar” targets in the make file.

For the maze shown above, we get the following results (the '2's are the resulting path.):

Queue search from (1,-1) to (10,10):

Starting point is not inside the maze!

Search canceled.

Stack search from (0,0) to (1,41):

Ending point is not inside the maze!

Search canceled.

Rat (stack) searching from (7,0) to (6,14):

000100000010000

000100100000000

002222211111111

222111221001000

222221122100100

110221102100000

011221102122222

201221102122222

211221002222222

221220111111111

021210000000000

021210001000000

022210010000000

I've traveled through **89** square(s). The path contains 58 square(s)

Multiple rats searching from (7,0) to (6,14):

**000100000010000**

**000100100000000**

**002222211111111**

**002111221001000**

**002001122100100**

**112201102100000**

**011201102100002**

**201201102100002**

**211201002222222**

**221200111111111**

**021210000000000**

**021210001000000**

**022210010000000**

We've traveled through 134 square(s). The path contains 40 square(s)

Smart rat searching from (7,0) to (6,14):

000100000010000

000100100000000

002222211111111

002111221001000

002221120100100

112221120100000

011221122122222

201201102120000

211201002220000

221200111111111

021210000000000

021210001000000

022210010000000

I've traveled through 48 square(s). The path contains 44 square(s)

• Your submission MUST run on thunder.cise.ufl.edu. That is where it will be run and graded.

The simplest syntax of the tar command is as follows.

tar cvf (tar\_file\_name) (file 1) (file 2) (file 3)...

To extract the contents of a tar file:

tar xvf (tar\_file\_name)

As before, use the Makefile to create the tarfile.

It is STRONGLY recommended you verify your submission is successful by downloading it from Sakai (into a separate directory), untarring it, running make (on both targets)

**Notes:**

**1. You may assume the maze will always have 15 columns.**

**2. You’ll need to guard stack.h, stack.cpp, queue.h, and queue.cpp against multiple #include’s of Arraylist.h. Arraylist.h doesn’t do it.**

**3. Do not modify any of the code provided.**