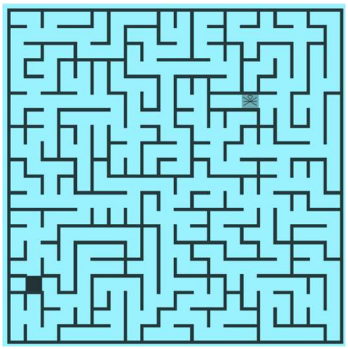
# 24-780 B—Engineering Computation

Assigned: Wed. Nov. 8, 2023
Due: Tues. Nov. 14, 2023, 11:59pm



## Problem Set 8: Ortho Maze Again

Once again, we'll build on a previous assignment, which means you have the choice of using your solution from PS07 or using mine (with some changes to project name and such to make it truly yours). This assignment has two main tasks, which can be done in *any order* since you do not need to finish one before the other can be tested.

### Task 1 (Find Shortest Path A*)

The ability to find the shortest path will again be implemented in Entity class, but this time you are asked to implement the A* (pronounced A-star) algorithm (which we'll discuss in lecture). You are to code the function Entity::findShortestPathAstar() and add the necessary code in MazeManager::manage() to run the algorithm when the user press the letter "A" (as opposed to "G"). Since our maze software has the ability to display the visited cells in the search algorithm, you can easily compare the performance of breadth-first search (BFS from PS07) and A* algorithms.

Note, you are asked to ADD a function, not to replace a function. Again, keep the old BFS algorithm intact in your code.

### Task 2 (Fully Editable Maze)

In PS07, you implemented the Aldous-Broder (aka, random walk) algorithm to create a perfect random maze (fully connected, no loops). I hope that, like me, the random maze left you a bit unsatisfied since you had no say in the little details of the maze. For this task, you are asked to add the ability to edit the maze by clicking on the area between cells with the *right mouse button*. It should work something like this:

- If there is a wall between the cells (i.e., cells are connected), the wall should disappear (i.e., cells become disconnected).
- Conversely, if there is NOT a wall between the cells (i.e., cells are disconnected), the wall should appear (i.e., cells become connected).

Look into how the functions Maze::getCellFromMouse() and Maze::changeCell() work and how they are called in MazeManager::manage() (e.g., when showing cell location and when setting start and end) for ideas about how to make this happen (i.e., how to determine which two cells are in question based on mouse click.

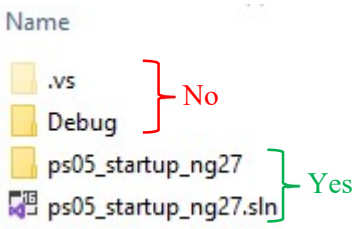
## Deliverables

Several files (zipped together):

MazeManager.h and .cpp
Maze.h and .cpp
Entity.h and .cpp

Note that you don't need to include all the "library files" (StringPlus, DrawingUtilNG, GraphicFont, etc.) unless you made some changes. Upload the zip file to the class Canvas page before the deadline (Tuesday, Nov. 14, 11:59pm).

Alternatively, if you are using Visual Studio, it may be easier to submit your entire solution rather than a collection of files. To do this, create a *zip file* of the whole project (the .sln file and the associated folder), being careful NOT to include the hidden folder called ".vs". This folder is used only to manage the IDE and is typically huge (100MB). Erasing or omitting it will just force Visual Studio to rebuild it when needed. The Debug and x64 folders should be kept out of the zip file too to avoid including executable files that some firewalls may disallow. *The name of the project should include your AndrewID*




## Learning Objectives

Use of classes and objects in C++.

Use of "large" dynamically allocated data models.

Increasing understanding of graphical data representation.

Mouse input handling.

Implementing algorithms developed by others.