Program 6: Recursion

Submit Assignment

Due Sunday by 11:59pm **Points** 13 **Submitting** a file upload **File Types** zip
Available after Sep 29 at 12am

FA20) Reminders:

1) Cells in a maze are connected in both directions. If you forget that, the default maze only works by luck. They happen to be connected in a way that leads to the exit.

2) You never go backwards! Ever ever. If copy three of the func calls a fifth copy on the north pointer, when the fifth copy finds it is stuck it just ends. Then the fourth picks up where it left off as if it never went that way. It doesn't make a sixth copy backwards.

3) Inside a single instance of the function, Current will NOT change. When you want to go north, you don't physically move your cell to the north. Current is the cell. If you want to do something with the cell to the north, they are tCurrent->mNorth. In a huge maze with 50 copies of the function on the call stack, each version would have a different Current pointer. Each cell has one.

5-13 #5) We want to escape a maze using recursion. We can't use a loop because it forks. We also need the actual resulting path.

With apologies to the book author, their answer is stupid. Instead of tracking the places you need to check, just try them. If they don't work out, then forget them. The difference is I'm using backtracking, and they are using breadth search. (We're doing tree theory stuff next week.)

pseudocode:

Return true if we are the exit

Push self on answer stack and mark self processed.

for each exit, Recurse on it

if any recurse returns true, yay! just keep returning true all the way back.

if none return true, pop us off answer stack and die quietly

Big extra credit: Write the destructor

Small Extra credit: Load a text file and make a maze out of it. A sample text file is in the handouts. Border is guaranteed to be solid X's.