CPSC 312

Assignment #1 - The Maze

Due no later than 2pm Tuesday, September 29th, 2015.

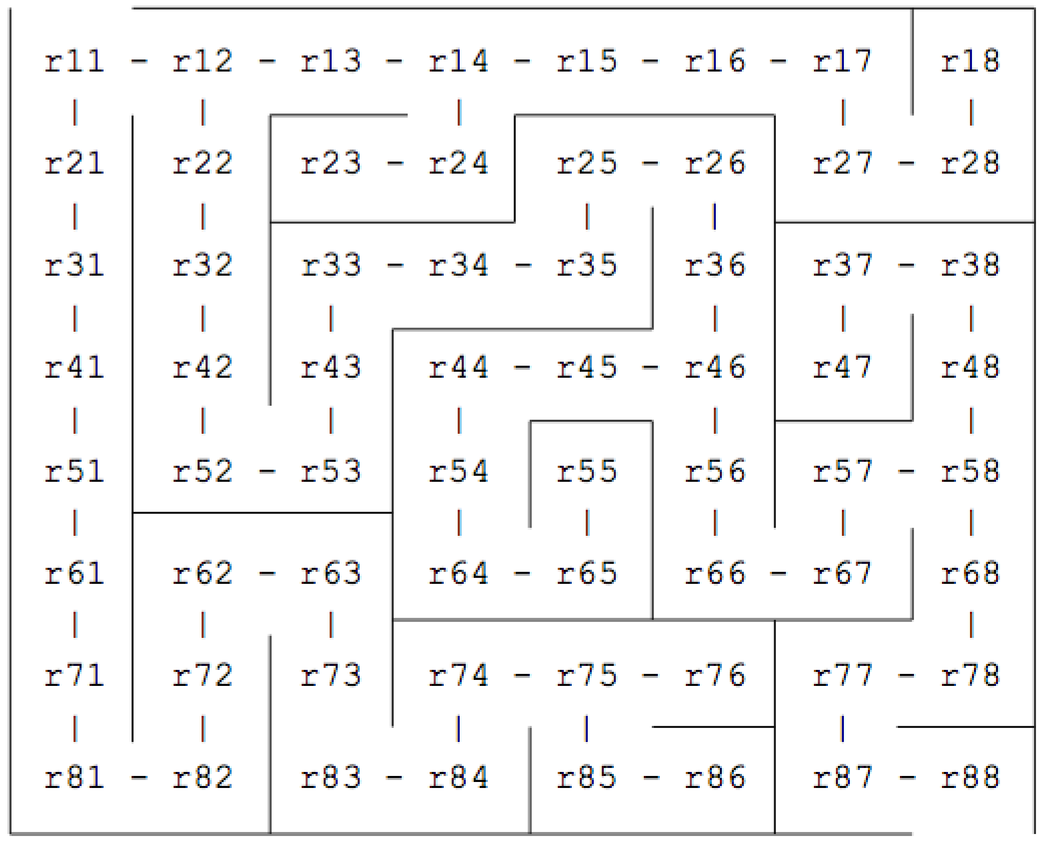
Please include your official full name, your student id as well as your ugrad id in your submission.

Collaboration Policy

The assignment will be handed in individually. However, you may choose to discuss your ideas with only one other student from the class, but even then you are not to share or take away anything from the discussion other than what's in your brains.

If you choose to have a discussion partner, you should mutually credit each other in your submissions, by including their name and student number in your submission.

The Maze



There is a path in this maze from the top left corner to the bottom right, from the point named r11 to the one named r88 (Can you see it?).

Download **maze.pl** from Piazza.
It contains all the facts for the maze shown above as a Prolog program. It also contains twelve different versions of the recursive procedure **path**, which finds paths between given points in the maze, if one exists.

Part A: Try each version of path.
Does that version find a proof that a path from r11 to r88 exists, or does the stack overflow?
Use SWI-Prolog's **time** predicate to see how much work Prolog did along the way. Record the output you observe for each version. If it did find a path and printed true, there is no need to press semicolon to get alternative answers. (2.5)

Part B: Now comment out one fact in maze.pl so that there is no path from r11 to r88. (The one you should comment out is indicated in the program). Use % to comment out a single line in Prolog.
Try all twelve versions of path again and record the outputs. (2.5)

Part C: Now answer these questions:

1) In Part A: Based on the different behaviours you observed from the different *path* procedures, what are some of the factors you see present in the code that can explain these differences? (2)
In particular,
1.a) how would you explain the difference in the performance of path01 and path02? (2)
1.b) what about the difference between path01 and path07? (2)
1.c) which one makes the highest number of inferences without causing the stack to overflow? why does it make so many inferences? (2)
1.d) which one makes the fewest number of inferences and succeeds? how? (2)

Bonus 1.e) Is the particular way the maze is represented in this program responsible for the difference in the number of inferences made by one set of path procedures compared to others? how? name one path procedure which benefits and one which suffers from this representation.
how can the representation be altered to reverse the effect (make it favor the other group instead)? (2.5)

Bonus 1.f) For path01 and path08, can you map out the steps taken by each through the maze to eventually reach the goal, including all the backtracks?

Example for path01: First it tries the path from r11 to r18, hits a wall and backtracks to r14, then goes from there to point x, then backtracks to point y, and …. then reaches r88. hint: there is a built-in swi-prolog predicate that can help you figure this out! (2.5)

2) Why do some versions of the path procedure work in Part A and Part B, while others work only in Part A? (2.5)

3) Based on the patterns you've observed in this exercise, can you present any conclusions about the best way(s) to order clauses in recursive programming? (2.5)

When answering these questions, think in terms of how your Prolog interpreter works, and how rule order and goal order in each path procedure affect how its recursion works (or doesn't).