Tree search framework
A Scheme tree search framework search.scm is available on csserver in directory /home/hwang/cs430/project1/. As discussed in class this framework implements various data structures and algorithms presented in the textbook including a problem structure (page 62), a node structure (page 69), a queue structure (page 71), and the tree search, node expansion, and solution functions (page 72). In addition, there are example implementations of the breadth-first-search and depth-first-search queues and the Romanian map problem.

As noted in class, each search technique is embodied in the fringe queue implementation that determines which node in the fringe is returned by the first function (and thus is the first node expanded in the fringe). For breadth-first search, the fringe queue is a standard queue. For depth-first search, the fringe queue is a stack.

To find a solution to the Romanian map example in the textbook (pages 62-64) evaluate the expression:

> (tree-search example-rom (make-bfs-fringe))

Different versions of the Romanian map problem can be created using the make-romanian-map-problem function. Just be sure to an empty fringe queue when you try to solve them, which can be done by creating a new one for each search. (Note that the DFS implementation does not avoid repeated states, so it is not very interesting since it gets into loops easily. To stop an execution in DrScheme, there is a stop button in the upper right-hand corner of the toolbar.)

Assignment
1. Implement fringe queues for uniform-cost, depth-limited, and A* searches. Add whatever fields you need wherever you deem necessary. For example, the heuristic function to the problem structure. Make sure the constructor functions and calls are adjusted appropriately if you do this.

2. Add the straight line distance data (Figure 4.1 page 95) to the project to support A* search for and implement the straight-line-distance heuristic function for the Romanian map problem. The SLD data may be added to the project in whatever manner you deem appropriate.

3. Implement a problem structure and a make-problem function for the missionaries and cannibals problem of Exercise 3.9 on page 90. This requires that you define a state representation, a goal-test function, a path-cost
function, and a successors function. (The step-cost and heuristic functions simply return 1, since we're only interested in the number of steps.)

4. Explore using the different search algorithms on the Romanian map problem and the missionaries and cannibal problem. Compare the algorithms (e.g., number of nodes expanded, optimality of solution, etc.) and argue for which is the most appropriate.

**What to turn in**
Submit your **well-commented** program file by email to the instructor. Submit a hardcopy printout of the your program file along with instructions on how to run the programs and your discussion for part 4.