Answer all questions. Test is open book and open notes. No talking or sharing of notes/books.

Please read all questions carefully. Several have detailed descriptions but require fairly brief answers.

1. (15 points) Unify the following expressions:

   (a) *(?x a b)* and *(b a ?y)*

   (b) *(a ?x ?y d)* and *(?x b d ?y)*

   (c) *F(?a, G(?a, ?b, c), c)* and *F(G(?x), G(G(k), c, ?y), ?y)*

2. (25 points) Hex-Pawns is a simple game played as follows:

```
 O | O | O         O | O | O          O | O | O
-----------       -----------        -----------
   |   |             |   |              |   | X
-----------       ---------X-        -----------
 X | X | X         X | X | ^          X | X |
```

   - The board is a 3x3 matrix;
   - Initially each player has three pawns in the back row;
   - Pawns may move forward one square, or capture (like a chess pawn) by moving diagonally;
   - The game is over if:
     (a) A pawn reaches the back row (the player wins);
     (b) A player has no move (the player loses).

   Assume X had the first move and moved the right-most pawn forward one position (as shown above). Draw the complete game tree and use it to show whether: X wins, O wins, or it cannot be determined.

3. (20 points) Show the effect of $\alpha - \beta$ pruning on the following MIN-MAX tree.

4. (15 points) A plan for a task is to be found using the situation calculus. The goal is to produce a situation in which certain facts are true. Actions, represented by functions, are allowed to accomplish this. Axioms are given for each action that state which values change by preforming an action in a given state.

   (a) Describe the *frame problem* for the situation calculus in general terms (ONE PARAGRAPH OR LESS). State *why* it is a problem, *what* frame axioms are, and to what extent frame axioms constitute a solution to the frame problem (copying an answer verbatim from the book will be marked as incorrect).

5. (25 points) Planning, STRIPS-style:
   Robbie is a robot that pushes train cars around on a side track to rearrange them into a desired order (as shown below). The side track consists of seven segments labelled 1 to 7 and connected as shown. Only one car at a time mat be on a segment of track, and Robbie can only push one car at a time.

   START STATE GOAL STATE

   To push a car from track segment t_1 to segment t_2, segment t_2 and all other in-between segments must already be cleared of cars. For example, in the initial state illustration, the only action available to Robbie is to push the wood car W from segment 1 to either segment 2 or segment 3 (since they are clear). In order to push W to segment 6, Robbie would have to clear segments 4 and 5. Robbie cannot push cars onto the main track.

   Initially there are four cars on the side track in the positions shown in the illustration (W ==wood car, P ==passenger car, E ==engine, C ==caboose). The initial state can be described by:

   Initial State: AT(W,1), AT(P,4), AT(E,5), AT(C,7), CLEAR(2), CLEAR(3), CLEAR(6)

   AT(c,t) means that car 'c' is on track segment 't' and CLEAR(t) means that track segment 't' is clear. These are the *only* predicates that can be used in this problem.

   (a) (10 points) Suppose the goal is to arrange the cars into a train as pictured in the illustration above. Write a single predicate logic formula containing as few literals as possible that expresses this goal in a form suitable for use in a STRIPS-like or POP-like planning system.

   (b) (10 points) Assume robbie as available a set of STRIPS-style operators: "MOVE(c,i,j)" means "move car c from track segment i to track segment j" where c is a variable but i and j are constants. There is one such move for each and every possible movement, e.g., MOVE(c,1,2), MOVE(c,1,3), MOVE(c,1,4), ..., MOVE(c,7,6), etc.

   Write the three parts of the *single* STRIPS-form operator "MOVE(c,7,2)."

   (c) (5 points) Using only the primitive robot actions as enumerated above, show how a simple STRIPS-like planner (i.e., situation-space, regression planner) would devise a plan for Robbie to transform the initial state to the given goal state. Do this by drawing *only* the sequence of goal stacks on a path to a solution; write beneath each goal stack the evolving solution sequence at that point. It is *not* necessary to show goal states or evolving plans for parts of the search graph not on the solution path.