1. Write a set of Prolog grammar rules that describe commands related to moving around in an area that includes a shed and a tent in a large field. The grammar should accept queries of these forms:

- "go", followed by a direction (north, south, east, or west), optionally followed by an adverb describing speed (quickly, rapidly, or slowly).
- "enter", followed by the, followed by a noun (shed or tent).
- "exit", followed by the, followed by a noun (shed or tent).

Assume that the input to the grammar will be lists of words in lower case letters, with no punctuation: [go, north], for example. (This is the standard input form for Prolog grammars.)

Thus, for example, your grammar should accept all of these sentences:

- [go, south]
- [go, west, slowly]
- [enter, the, shed]
- [exit, the, tent]

but should reject all of these sentences:

- [enter, the, tent, quickly]
- [go, uphill]
- [shed, north, go]

The rules for noun, direction, and speed_adverb are given below. Add the remaining rules needed for the grammar.

```
noun --> [shed].
noun --> [tent].
direction --> [north].
direction --> [south].
direction --> [east].
direction --> [west].
speed_adverb --> [quickly].
speed_adverb --> [rapidly].
speed_adverb --> [slowly].
```
2. Write a Prolog predicate to represent a left-slide tile move in a sliding block puzzle, as described below:

   The sliding block puzzle we studied in class contains 3 black tiles and 3 white tiles which can slide or hop within a row of 7 spaces. A puzzle state will be represented in Prolog as a length-7 list containing the symbols b (representing a black tile), w (a white tile), and space (the empty space). For example,

   \[
   \text{BBB WWW}
   \]

   will be represented as the list \([b,b,b,\text{space},w,w,w]\).  

   One legal move in this puzzle is to select the tile immediately to the right of the empty space and slide it one square to the left. For example, beginning with this puzzle state,

   \[
   \text{BBB WWW}
   \]

   if we slide the first W tile to the left, we will get the puzzle state

   \[
   \text{BBBW WW}
   \]

   Write a Prolog predicate `slide_left(+OldPuzzleState, -NewPuzzleState)` which takes a length-7 list describing a puzzle state as its first argument (OldPuzzleState), and succeeds with NewPuzzleState set to the result of sliding a tile one square to the left, if this move is possible. If it is not possible to slide a tile one square to the left, `slide_left` should fail.

   For example,

   \[
   \text{slide_left([b,b,b,\text{space},w,w,w], NewState) should succeed with NewState set to [b,b,w,\text{space},w,w].} \\
   \text{slide_left([space,w,b,w,w,b,b], NewState) should succeed with NewState set to [w,space,b,w,b,b].} \\
   \text{slide_left([w,w,b,w,b,b,\text{space}], NewState) should fail.}
   \]
The certainly factor formulas used by MYCIN are listed on the last page of this exam, for your reference. Show all your work for the problem below, so that I can give partial credit in case of math errors.

3. The following MYCIN-style rules identify some kinds of program bugs, using certainty factors:

   - Segmentation fault & uses dynamic variables $\rightarrow$ dangling pointer (0.5)
   - Unpredictable output & uses dynamic variables $\rightarrow$ dangling pointer (0.5)
   - Uses loops & takes a long time to finish $\rightarrow$ infinite loop (0.7)

   a) If the program uses dynamic variables (CF = 1), has unpredictable output (CF = 0.2), and has a segmentation fault (CF = 0.8), it has a dangling pointer with what certainty factor?

   b) If the program uses loops (CF = 1) and takes a long time to finish (CF = 0.9), it has an infinite loop with what certainty factor?

4. Select any 3 of the expert systems listed below and briefly (in a word or phrase) describe their area of expertise.

   - MYCIN
   - DENDRAL
   - PROSPECTOR
   - AM
   - MACSYMA
   - XCON

5. One of your friends, hearing that you’re taking a class in artificial intelligence, says "Computers can never be intelligent — they only do what they’re programmed to do." How would you reply to your friend? (You may agree or disagree with your friend, but give good reasons for the position you take. What would it require for a machine to be "intelligent", and do you think a collection of hardware and software can actually achieve intelligence? Why or why not?)
MYCIN Certainty Factor formulas

Certainty Factor of combined rule premises

\[ CF(P1 \text{ and } P2) = \min(CF(P1), CF(P2)) \]
\[ CF(P1 \text{ or } P2) = \max(CF(P1), CF(P2)) \]

Combining evidence from multiple rules:

If

Rule 1 gives support \( CF(R1) \) and
Rule 2 gives support \( CF(R2) \)

Then the combined certainty factor is:

- a) \( CF(R1) + CF(R2) - CF(R1) \times CF(R2) \) if both \( CF(R1) \) and \( CF(R2) \) are positive
- b) \( CF(R1) + CF(R2) + CF(R1) \times CF(R2) \) if both \( CF(R1) \) and \( CF(R2) \) are negative
- c) \( \frac{CF(R1) + CF(R2)}{1 - \min(|CF(R1)|, |CF(R2)|)} \) otherwise