UNIVERSITY OF OTTAWA
FACULTY OF ENGINEERING
SCHOOL OF IT AND ENGINEERING

CSI 2165 B

Midterm
February 28, 2006
8-10 am

Examiner:
Dr. Diana Inkpen

<table>
<thead>
<tr>
<th>Name</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Student Number</td>
<td></td>
</tr>
</tbody>
</table>

Total marks: 45
Duration: 110 minutes
Total Number of pages: 10

Important Regulations:
1. No calculators are allowed.
2. A student identification card (or another photo ID) is required.
3. An attendance sheet shall be circulated and should be signed by each student.
4. Please answer all questions on this paper, in the indicated spaces.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>10</td>
<td>12</td>
<td>4</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>45</td>
</tr>
</tbody>
</table>
A. [5 marks]
Which of the following are syntactically correct Prolog objects? If yes, identify the types of object they are (atom, number, variable, structure). If not, use a “No” as an answer.

<table>
<thead>
<tr>
<th>Term</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ab13</td>
<td>Atom</td>
</tr>
<tr>
<td>2. const3</td>
<td>Variable</td>
</tr>
<tr>
<td>3. A(B)</td>
<td>No</td>
</tr>
<tr>
<td>4. [1, [2], f(a)]</td>
<td>Structure (list)</td>
</tr>
<tr>
<td>5. country(name('Canada'))</td>
<td>Structure</td>
</tr>
</tbody>
</table>

B. [10 marks]
Show the variable instantiations (the values of X, Y and Z) if the matching between the term in the first column and the term in the second column succeeds. Use “No” otherwise.

<table>
<thead>
<tr>
<th>Term1</th>
<th>Term2</th>
<th>Instantiations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>Term1</td>
<td>Term2</td>
</tr>
<tr>
<td>1. [1, 2, 3]</td>
<td>[X</td>
<td>Y]</td>
</tr>
<tr>
<td>2. [1</td>
<td>[2,3]]</td>
<td>[X, Y]</td>
</tr>
<tr>
<td>3. [[a, c], g(Y)]</td>
<td>[X, g(b)]</td>
<td>X = [a, c]</td>
</tr>
<tr>
<td>4. [f(a), g(b)]</td>
<td>[X</td>
<td>Y]</td>
</tr>
<tr>
<td>5. f([1,2], [a,b,c])</td>
<td>g(X,Y)</td>
<td>X = No</td>
</tr>
<tr>
<td>6. [X</td>
<td>Y, Z</td>
<td>[a]]]</td>
</tr>
<tr>
<td>7. [1, 2</td>
<td>[3]]</td>
<td>[X</td>
</tr>
<tr>
<td>8. [1, 2]</td>
<td>[X</td>
<td>Y, []]</td>
</tr>
<tr>
<td>9. [1, [2, 3]]</td>
<td>[</td>
<td>[X</td>
</tr>
<tr>
<td>10. a([X, [Y, a(X)])]</td>
<td>a([[a</td>
<td>[[X</td>
</tr>
</tbody>
</table>
C. [12 marks]

Read the following programs and provide the answer to the query (only the first answer that SWI-Prolog would produce).

| 1. Program | prog_1([],[]).
|            | prog_1([X|T], [X,X|T1]) :- prog_1(T, T1).
| Query      | ?- prog_1([a, b],R).
| Answer     | R = [a, a, b, b] |

| 2. Program | prog_2(L,R) :- acc(L, 1, R).
|            | acc([],A,A).
|            | acc([H|T],A,R):- Al is A*H, acc(T,Al,R).
| Query      | ?- prog_2([1,2,3,4],R).
<p>| Answer     | R = 24  multiplies all elements |</p>
<table>
<thead>
<tr>
<th></th>
<th>Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.</td>
<td>( \text{prog}_3([], Y, [Y]). )  \ ( \text{prog}_3([H</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \text{prog}_3([a,b,c], 5, R). )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( R = [a, b, c, 5] )  \ adds element at end of list</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.</td>
<td>( \text{prog}_4(X, [X]). )  \ ( \text{prog}<em>4(X, [</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(?- \text{prog}_4(X, [a, b, c, d]). )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( X = d, ) find the last element</td>
</tr>
</tbody>
</table>
| 5. Program | prog_5([X|Xs], 1, Xs).
|            | prog_5([Y|Xs], K, [Y|Ys]) :-
|            |     K > 1,
|            |     K1 is K - 1,
|            |     prog_5(Xs, K1, Ys).
| Query      | ?- prog_5([a, b, c, d], 2, R).
| Answer      | R = [a,c,d], remove element at given position |

| 6. Program  | prog_6(L, X):-
|             |     append(L1, [X] L2), L),
|             |     length(L1, Y),
|             |     length(L2, Y).
| Query       | ?- prog_6([a, b, c, d, e], X).
| Answer       | X = c, middle element |
D. [4 marks]

Given the two predicates `del1` and `del2`, fill the table below with the SWI-Prolog’s response to the corresponding query. The predicates are similar, but the clause order is different. The first argument is an element, the second is a list, and the third is a list where the result is stored.

Note that you have to include all answers by assuming you hit “;” instead of “ENTER”.

```
del1(_, [], []).
del1(X, [X | Tail], Tail_1):- del1(X, Tail, Tail_1).
del1(X, [Y | Tail], [Y | Tail_1]):- del1(X, Tail, Tail_1).

del2(_, [], []).
del2(X, [Y | Tail], [Y | Tail_1]):- del2(X, Tail, Tail_1).
del2(X, [X | Tail], Tail_1):- del2(X, Tail, Tail_1).
```

<table>
<thead>
<tr>
<th>Query</th>
<th><code>del1(1, [2,1], P)</code></th>
<th><code>del2(1, [2,1], P)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>All answers</td>
<td>P = [2] ;</td>
<td>P = [2, 1] ;</td>
</tr>
<tr>
<td></td>
<td>P = [2, 1] ;</td>
<td>P = [2] ;</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
E. [7 marks] Facts and rules in Prolog

In the kingdom Logiquogne the penal code contains the following laws:

*All the thieves are criminals.*
*All the persons who help a criminal are also criminals.*
*The adult criminals go to prison.*
*The minor criminals go to a correction house.*

1. The kind asks you to write a Prolog program to replace the judge in the courthouse. Translate the laws into Prolog, using the following symbols for predicates:

thetaif(X): X is a thief  
criminal(X): X is a criminal  
helps(X,Y): X helps Y  
adult(X): X is an adult  
minor(X): X is a minor  
prison(X): X goes to prison  
correction(X): X goes to the correction house

2. Using the following database:

```prolog
thief(thomas).
thief(arsene).

minor(thomas).
minor(philibert).
minor(paul).
minor(sara).

adult(arsene).
adult(mireille).
adult(caroline).
adult(charles).

helps(philibert,mireille).
helps(paul,arsene).
helps(paul,charles).
helps(arsene,philibert).
helps(mireille,sara).
helps(charles,caroline).
helps(caroline,thomas).
```

2.1. What Prolog query do you need to ask for finding all the persons who go to prison?
2.2. What are the Prolog answers if you press ; after each solution?

2.3. What Prolog query do you need to ask for finding all the persons who go to the correction house?
2.4. What are the Prolog answers if you press ; after each solution?
Solution  
1. Rules:  

   criminal  
   criminal(X) :- thief(X).  
   criminal(X) :- helps(X,Y), criminal(Y).  

   prison  
   prison(X) :- criminal(X), adult(X).  

   correction  
   correction(X) :- criminal(X), minor(X).  

2. Queries and answers:  

   ?- prison(X).  
   X = arsene ;  
   X = charles ;  
   X = caroline ;  
   No  

   ?- correction(X).  
   X = thomas ;  
   X = paul ;  
   X = paul ;  
   No
Write a predicate named \texttt{rem\_first} that has three arguments: a list, a number, and an output list. The predicate removes a given number of elements from the front of the given list. Assume the number is less or equal to the length of the list.

Examples:
?- \texttt{rem\_first([a,b,c,d,e,f,g], 4, L)}.
L = [e,f,g] ;
No

?- \texttt{rem\_first([a,b,c,d,e], 1, L)}.
L = [b,c,d,e] ;
No

\begin{verbatim}
rem\_first(L, 0, L).
rem\_first([_ | T], N, T1):- N1 is N-1, rem\_first(T, N1, T1).
\end{verbatim}
G. [4 marks]

Write a predicate named `pairs` that has three arguments: a first list, a second list, and a third argument. The predicate returns in the third argument a list of all the possible pairs of elements from the two given lists.

Example:
```prolog
?- pairs([1, 2, 3],[a, b],L).
L = [[1, a], [1, b], [2, a], [2, b], [3, a], [3, b]] ;
No
```

```prolog
pairs([], _, []).
pairs([H|T], L2, R):- p_helper(H, L2, R1), pairs(T, L2, R2), append(R1, R2, R).
p_helper(_, [], []).
p_helper(E, [H|T], [[E, H]|T1]) :- p_helper(E, T, T1).
```