CSCE 580 Spring 2003
SECOND MIDTERM EXAM
Wednesday 2003-04-22—Two Pages, Closed Book
50 points

1 Prolog—50 points

1. If it exists, give the most general unifier for:

   (a) (2 points) `f(s(s(s(X))))` and `f(s(s(s(s(Y)))))`. **Answer:** {X \ s(Y)}

   (b) (3 points) `knows(john,X)` and `knows(Y,mother(Y))` **Answer:** {Y \ john, X \ mother(john)}

   (c) (3 points) `q(Y,g(a,b))` and `q(g(X,X),Y)` **Answer:** Does not exist

2. Consider the following three definitions of the ancestor relation[1] (called "predecessor" in your textbook.) Assume that the parent relation is already defined.

```
anc1( X,Y) :- parent( X,Y).
anc1( X,Y) :- parent( X,Z), anc1( Z,Y).

anc2( X,Y) :- parent( X,Y).
anc2( X,Y) :- anc2( Z,Y), parent( X,Z).

anc3( X,Y) :- parent( X,Z), anc3( Z,Y).
anc3( X,Y) :- parent( X,Y).
```

   (3 points) Which ones of the three definitions are both declaratively and procedurally correct?

   Answer: the first one and the third one are correct.

   (3 points) Provide a query on which the other definition fails (e.g., by getting into an infinite loop).

   Answer: The second one fails (infinite loop) on `?-anc2(a,b).`, where a and b are not in the parent relation.

   (1 point) Which one of the two correct definitions is more efficient?

   Answer: anc1.

[1] For the purpose of this exam, "relation" and "predicate" are used interchangeably.

3. (5 points) Define the predicate `reverse( List, ReverseList)`, with the obvious meaning. Assume that the predicate `conc(L1,L2,L3)` (list concatenation, also known as `append`) described in class is already defined. **Answer:** rev([],[]). rev([H|T],Z) :- rev(T,T1), append(T1,[H],Z).

4. (5 points, exercise 3.5[B], modified) Define the predicate `palindrome( List)`. A list is a palindrome if it reads the same in the forward and backward direction. For example, `[m,a,d,a,m]`. Assume that `List` contains only constants. **Answer:** palindrome([]). palindrome(L) :- reverse(L, L1), L = L1. Also (Bratko): palindrome([]). palindrome([A]). palindrome(L) :- append([First|Middle], [First], List), palindrome(Middle).

5. (5 points) Define the predicate `delete( Element,List1,List2)` that holds when List2 is List1 with one occurrence of List2 deleted.

   Answer: `del( X,[X|L],L).`
   `del( X,[ Y|Tail],[ Y|Tail1]) :- delete( X,Tail,Tail1).`

6. (5 points) The `append_dl` relation, `append_dl( X-Y,Y-Z,X-Z).`, defines append (viz. concatenate) with difference lists.

   Explain how it works by showing how terms are matched (viz. unified) for the goal `?-append_dl([x,y|A]-A,[z,w|C]-C,Result).`

   Answer: `{X / [x,y|A]}`
   `{Y / A}`
   `{A / [z,w|C]}`
   `{Z / C}`
   verb+Result / X-Z+, i.e., `{Result/[x,w|A]-C}`, i.e. `{Result/[x,y,z,z|C]-C}`

7. Define a relation `notIn(X,L)` that holds if X is not in L (5 points). What does your program answer to the following three queries?
   (1 point)`?- notIn(a,[]).`
   (1 point)`?- notIn(a,[b,c,d]).`
   (1 point)`?- notIn(X,[a,b]).`
   (1 point)`?- notIn(a,[X,b]).`
   (1 point)In which way is this question not specified precisely?

   Answer:

```
1 ?- [user].
|    notIn(X,[]).
Warning: Singleton variables: [X]
|    notIn(X,[Y|L]) :- X \= Y, notIn(X,L).
|    ^D
% user compiled 0.00 sec, 588 bytes

Yes
2 ?- notIn(a,[]).
```

```
Yes
3 ?- notIn(X,[a,b]).

No
4 ?- notIn(a,[X,b]).

No
5 ?- notIn(a,[b,c,d]).

Yes
```

What does "is not" mean? Does not unify? Is textually different?, etc.

8. (5 points, exercise 5.5 [B]) Let S1 and S2 be two lists representing sets. Define a relation `set_difference( X,Y,Z)` that holds if Z is the set difference of X and Y (i.e., Z contains the elements of X that are not in Y). Assume that `member\2` has already been defined. Hint: use cut to distinguish the case in which the first element of the first list is a member of the second list from the case in which the first element of the first list is not an element of the second list.

   Answer: `set_difference( [],_,[]).`

   ```
   set_difference([X|L1],L2,L) :- member(X,L2),!,set_difference(L1,L2,L).

   set_difference([X|L1],L2,[X|L]) :- set_difference(L1,L2,L).
   ```