Name_____ NetID _____

CS 314 Practice Exam 1

0. For each of the following pairs of prolog fact and goal, say whether they match (i.e. unify), and if so what bindings will be made.

	fact	goal	
A.	tv(sony).	tv(Sony).	matches, Sony = sony
B.	tv(Sony).	tv(Tv).	matches, Tv = Sony
C.	bro(Bro, Bro).	bro(bob, X).	matches, X = Bro = bob
D.	eats(Cat, fish).	feeds(joe, X).	~~matches~~, does not match
E.	last(a, [b, c]).	last(X, [X | T]).	does not match

The following applies to problems 1 through 4

- You only need to worry about the first answer prolog will return, not about what will happen if the user types a ;
- Each of these questions have answers that take at most 3 clauses. However, you may use as many clauses as you wish.
- You may define helper predicates if you wish.
- Your answers may use any built in prolog predicate, including
  - **member**(Elt, List), which is true if Elt is a member of List.
  - **append**(L1, L2, Lappended) which is true if appending L1 to L2 results in Lappended.

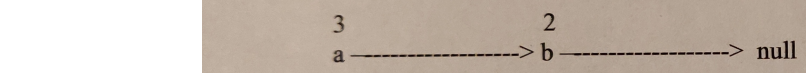
1. Define the predicate msaller_of(N1, N2, L). N1, N2, and L are numbers. If N1 is smaller than or equal to N2, L is N1, otherwise L is N2. E.g, smaller_of(4, 5, L) binds L to 4.

```
smaller_of(X,Y,X) :- X<=Y.
smaller_of(X,Y,Y) :- X>Y.
```

2. Define the predicate square_all(Numbers, Results) which multiples each member of Numbers by itself, and binds Results to a list of the results. E.g., square_all([8, 2], L) binds L to [64, 4].

```
square_all([],[]).
square_all([N1|N],[R1|R]) :-
    R1 is N1 * N1, square_all(N,R).
```

3. Suppose you represent a linked list as a functor wtih two arguments: the data at that node, and the next node the current node points to, with a 'null' node being an empty list. The list drawn below:



... would be represented as:

```
node(3, node(2, [])).
```

The prolog predicate listSum(LinkedList, Sum) binds Sum to be the arithmetic sum of all values in the linked list LinkedList, for example:

```
listSum( node(3, node(2, [])), Sum),
```

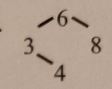
... would bind Sum to 5.

Define the rule listSum.

```
listSum([],0).
listSum(node(V,Node),Sum) :-
    listSum(Node,Sum1), Sum is V+Sum1.
```

4. Suppose we represent a binary search tree using a Prolog list with three elements: the node data, the left subtree and the right subtree. An empty tree is represented by null.

So: [6, [3, null, [4, null, null]], [8, null, null]] represents this tree



Write the predicate bstNotFound(BST, Value), where BST is a binary search tree as defined above, and Value is a number, which is true if Value does not appear in BST. For example, if 'BST' is bound to the list above; bstNotMember(BST, 10) is true ("yes") and bstNotMember(BST, 8) is false ("no"). Make sure to exploit the implicit ordering of data values as you scan the binary search tree.

```
bstNotFound([Data, null, null], Value) :-
    Data \= Value.

bstNotFound([Data, Lst, _], Value) :-
    Value < Data, bstNotFound(Lst, Value).
bstNotFound([Data, _, Rst], Value) :-
    Value > Data, bstNotFound(Rst, Value).
```