Name________________________

CS 314 Practice Exam 2

Fall 2013 – Answers

rev 2 – answer for question I added

I. Finish the definition below of the macro and-not, so that (and-not x y) is the equivalent of (and x (not y)). Note that and-not should have the same kind of "short cut" behavior as and and or have, and as x && y has in Java: if x is false, it should not evaluate y, but should just return false.

```
(define-syntax and-not
    (syntax-rules ( )
      ((_ x y)

       (and x (not y))




         )))
```

II. Fill in the following prolog predicates.

A. inOrder(List). Assume List is a list of numbers; inOrder(List) is true if and only if the numbers in List are in increasing order. If List has 0 or 1 element it is in order.

inOrder([ ]) .

inOrder([_]) .

inOrder([N1, N2 | T]) :- N1 < N2, inOrder([N2 | T]).

B. suffix(L1, L2). Assumes L1 and L2 are lists. It is true if L2 is a tail of L1. E.g., suffix([x, a, b], [a, b]) is true, as is suffix([x, y, a, b], [a, b]), suffix([x, y, z, a, b], [a, b]), and so on.

suffix([H | T], T).
suffix([H | T], T1):- suffix(T, T1).

III. For each of the following pairs of fact and goal, say whether they will unify, and if so with which bindings

	fact	goal	
1.	foo(Bar, baz).	foo(baz, Bletch).	Bar = baz, Bletch = baz
2.	foo(a, b)	fie(X,Y)	not unify
3.	foo(a, fie(Y, X))	foo(X, fie(a,a))	X=Y=a

III. the predicate odds(All, Odds) is true if Odds and All are lists and Odds contains the 1st, 3rd, 5th, etc, elements of All (in that order), Hint: [a, b | c] is a list whose first two elements are a and b, and whose tail after b (i.e. the cddr) is c. odds([],[]) is also true. Define odds:

```
odds([],[]).

odds( [ X], [X]).

odds( [H1, H2 | T], [H1 | T1] ):- odds(T, T1).
```

IV. What is the translation into predicate calculus of the prolog rule:
cousin(X,Y):- grandparent(X, G), grandparent(Y, G), nonsibling(X, Y).

for all X, Y cousin(X, Y) if there exists a G such that grandparent(X, G) and grandparent(Y, G) and nonsibling(X, Y)

V. What will this print for the query foo(X, Y). ? (The predicate write simply prints its argument.

```
foo( 1, 2):-write(12), 1<1.
foo(X, 2):- fie(X), write(x2), 2<1.
foo(1,Y):- write(y1).

fie(a):-write(a).
fie(b):-write(b).
```

```
12 a x2 b x2 y1
X = 1
```

VI. Define the predicate insertInOrder(Lst, Num, Res), where Lst is a list of numbers in ascending order and Res is the result of inserting Num in its correct place in Lst. E.G., insertInOrder([1, 3, 6, 9], 5, [1, 3, 5, 6, 9]) is true.

```
insertInOrder([ ],N,[N]).
insertInOrder([H|T],N, [N, H | T]):- N<H.
insertInOrder([H|T],N, [H|T1]):- N>=H, insertInOrder(T,N,T1).
```

VII. Given the following code, what will the query vacation(A) print? Hint: If the variable V has value a, write([fun, V]) prints [fun, a]. fail is a predicate that always fails.

```
vacation(Activity):-
     fun(Activity),
     write([fun, Activity]),
    cheap(Activity),
     write([cheap, Activity]),
     !,
     fail.
fun(Activity):- speed(Activity, S), S > 50.
fun(Activity):- outdoors(Activity).
speed(skiing, 75).
outdoors(skiing).
outdoors(hiking).
cheap(hiking).
```

```
[fun,skiing][fun,skiing][fun,hiking][cheap,hiking]
false
```