CS140-Fall 2006
Midterm Exam
October 25
You may use books or notes but no computers are allowed

Additional recommendations:
Students should occupy staggered seats and should not communicate with each other. If you find that a question contains ambiguities state clearly your interpretation in the exam book.

Question 1 (5 points)
Consider the following program:
f(0, zero). f(s(0), one). f(s(s(0), two).
f(s(s(s(X))), N):- f(X,N).
What is the result of the query **f(s(s(s(s(s(0))))), R)**?
What is a good name for the predicate "**f**" ?

Question 2 (5 points)
Given a list of integers L write the Prolog predicate **ordered (L)**, that checks if the list contains ordered integers, smallest in the beginning of L, largest at its end. Note that the integers **are not** in the successor s(X) notation, but simple numbers like 20, 34,…etc.

Question 3 (10 points)
Write a (parameter-less) predicate **maketable** that asserts all the elements of a multiplication table expressed in the form of 100 facts of the type:
product (X, Y, Z).
such that Z is X*Y for all X and Y between 0 and 9
Your program **has to** utilize the nondeterministic features of Prolog by invoking "**member(V, [0,1,2,…,9])**" and the predicate and **asserta**.

Question 4 (Total: 20 points)
Part 1 (5 points)
Write the DCG's for recognizing $a^n c\, b^n$ with $n >= 0$
In other words complete the following:
S -- >
Show the query to parse the string [a,a,c,b,b]
Part 2 (5 points)
Provide the translation of your answer to Question 4 into Prolog clauses.
Part 3 (10 points)
Change the DCG's in Question 4 so that you will be able to count the number of *n*'s using the successor function. In other words complete the following:
S(…) -- >
Show the query to parse the above string yielding the number of *n*'s.

Question 5 (10 points)
Assume you are given the following constraint program clause:
on_circle(p(X,Y), c(A,B, $(X-A)^2 + (Y-B)^2$)).
The above specifies **all points** (X,Y) lying on the perimeter of a circle whose center is the point of coordinates (A,B) and whose radius R is such that:
$R^2 = (X-A)^2 + (Y-B)^2$
Given the query:
? on_circle (p(7,1), C) , on_circle (p(0,2), C).
Explain the meaning of the query by sketching the geometric circles satisfying it.
Notice that the variable **C** is the same in both calls of **on_circle**.

Question 6 (Total 15 points)
Part 1 (10 points)
Write a program that tests if a Boolean formula expressed in terms is satisfiable.
In other words **satisfiable (Booleanterm)** is true if there is an assignment (true or false) to each of the variables in **Boolean term** that render it satisfiable (that is true).
Boolean term can contain embedded terms like:
and(X,Y), or(X,Y), not(X)
In other words you have to produce the clauses like:
satisfiable (and(X,Y)):- …
satisfiable (or(X,Y)) :- …
satisfiable (not(Z)) :- …
Obviously,
satisfiable (true).
Part 2 (5 points)
Write the query corresponding to a **tautology** (satisfiable for all combinations true and false for the variables).

Question 7 (20 points)
Unification in Prolog
unify(X,Y) succeeds if X unifies with Y.
Assume that you have the following built-in predicates
var(X) is true if X is a variable
constant (X) is true if X is an atom, i.e., a constant.
nonvar(X) is true if X is not a variable
compound(X) if X is a term
functor(X, F, N) called with a term X , yields the functor name F, and the number N of arguments in X..
For example the call:
functor (add(2, mult(3,4)), Name, Nargs) yields Name = add Nargs =2
arg(N, X, ArgX) determines the Nth argument of a term X. For example
arg (2, add(2, mult(3,4)), U) yields the second argument of add: U=mult(3,4),

Hint:
Write the following predicates
unify (X,Y) , is true if the term X unifies with the term Y

termunify(X,Y) is called when **both** X and Y are terms and yields true if the terms are unifiable.
unifyargs(N, Arg1, Arg2) unifies the Nth (integer) arguments of the terms Arg1 and Arg2.

Question 8 (15 points)
Write a mini-solve in Prolog
Assume that the program rules are represented by program clauses of the type:
rule (Head, Tail). Furthermore it is assumed that predicates do not contain parameters

For example, the Prolog program:

a:-b, c.
b.
c.

is represented by the clauses:

rule (a, [b,c]). rule (b, []). rule (c, []).

The predicate **solve** should have two parameters: the first is the current list of goals the second is the list of goals that remain to be processed. Initially the latter is represented by the nil list.

For example the query is ?a,c. is represented by a call of **solve([a,c], []).**
You are entitled to use **append.**