CS 381: Practice Problems for the Final Exam

1. Which sentences can be derived with the following grammar?

s ::= **a** s **b** | s **b** | ε

a) ab
b) aab
c) aabbb
d) abb
e) aabb
f) ba
g) bb
h) aababb

2. Consider the following excerpt from a context-free grammar for expressions.

exp ::= *num* | *exp* + *exp* | *exp*++

Define the abstract syntax for the shown productions as a Haskell data type.

3. Consider the abstract syntax for the following expression language. The constant Zero denotes 0, and the operation Succ computes successors. The operation Sum takes a list of expressions as arguments and computes its sum. IfPos returns the value of the first expression if its value is greater than 0. Otherwise, it returns the value of the second expression. You may use the function sum :: [Int] -> Int in your solution.

```
data Expr = Zero
        | Succ Expr
        | Sum [Expr]
        | IfPos Expr Expr
```

Define the semantics of the expression language as a function sem of the following type.
sem :: Expr -> Int

4. Consider the abstract syntax of a language for describing movements on a two-dimensional grid. The meaning of JumpTo p is to immediately go to the position indicated by p, regardless of the current position. In contrast, the command UpBy i moves from the current position up by i units. The operation Right yields the position given by 1 unit to the right of the current position. Finally, Seq m m' first performs the move m and then the move m'.

```
type Pos = (Int,Int)

data Move = JumpTo Pos
          | UpBy Int
          | Right
          | Seq Move Move
```

Define the semantics of the language Move by completing the following function definition.

```
sem :: Move -> Pos -> Pos
```

5. Determine the type/behavior of the following expressions under static and dynamic typing.

a) if tail x==[] then 3:x else True
b) not (tail [x,True])
c) if x=3 then x+1 else not x

6. Consider the following abstract syntax for a language for nested pairs of booleans, that is, pairs that can contain booleans and other pairs. Fls and Tru represent boolean constants. The operation Pair constructs a pair from its arguments. We can extract the first component of a pair using First, and the operation Swap exchanges the first and second component of a pair.

```
data Expr = Fls
          | Tru
          | Pair Expr Expr
          | First Expr
          | Swap Expr
```

Which of the following expressions should be considered to be **not type correct** by a type checker for that language?

```
a) First (First (Pair Fls Tru))
b) Swap (Pair Tru)
c) First (Fls Tru)
d) Swap (Pair Tru (Pair Tru Fls))
e) Swap (Swap (Pair Fls Tru))
f) First (Swap (Pair Fls Tru))
```

7. Consider the following facts about the prices of items at different stores.

```
price(amazon,a,2).
price(amazon,b,3).
price(amazon,d,7).
price(bestbuy,a,3).
price(bestbuy,c,5).
price(walmart,a,4).
price(walmart,b,5).
price(walmart,d,6).
```

Note: Don't define any auxiliary predicates. Don't use any built-in/predefined predicates other than comparison or arithmetic operators

a) Write a Prolog goal to compute all stores that sell item "a" for less than 4.

b) Define the predicate beats/3 that holds for two stores and an item when the first store sells the item for less than the second store.

c) Define the predicate shop/3 that computes for a given store and a given list of items the total amount that one has to pay to buy all the items in the list at that store.

d) Define a predicate prefer/3 that holds for two stores and a list of items when you can buy the complete list of items for less at the first store than at the second store.

8. Illustrate the evolution of the runtime stack from line 3 up until after the assignment on line 8 under the parameter passing scheme Call-by-Reference. (Add the line number to each runtime stack.)

```
1 { int a := 2;
2   int b := 4;
3   int f(int c) {
4     a := c+3;
5     c := a-2;
6     return (a*c);
7   };
8   b := f(a);
9 }
```

9. Consider the following block. Draw the runtime stacks that result immediately after the statements on lines **6**, **3**, and **7** have been executed for both dynamic scoping and static scoping.

```
1 { int u := 1;
2    {  int g(int x) {
3           u := u+x;
4            return (u*2);
5       };
6       { int u := 2;
7             u := g(u+3);
8        };
9    };
10 } ;
```

10. Determine what the value of x is at the end of the following program, and show the development of the runtime stack under the different parameter passing schemes. (You should always assume *static scoping*.)

```
1  { int x;
2    x := 2;
3    int f(int y) {
4       x := y*2;
5       return y+1
6     };
7     x := f(x+3)-1;
8  }
```

a) Show the evolution of the runtime stack and the final value of x under *call-by-name*.
b) Show the evolution of the runtime stack and the final value of x under *call-by-need*.