CS421 Fall 2010 Midterm 2

Thursday, November 11, 2010

Name:	
NetID:	

- You have **75 minutes** to complete this exam.
- This is a **closed-book** exam.. You are allowed one 3inch by 5 inch card of notes prepared by yourself. This card is **not to be shared**. All other materials, besides pens, pencils and erasers, are to be away.
- Do not share anything with other students. Do not talk to other students. Do not look at another student's exam. Do not expose your exam to easy viewing by other students. Violation of any of these rules will count as cheating.
- If you believe there is an error, or an ambiguous question, you may seek clarification from myself or one of the TAs. You must use a whisper, or write your question out. Speaking out aloud is not allowed.
- Including this cover sheet and rules at the end, there are 18 pages to the exam. Please verify that you have all 18 pages.
- Please write your name and NetID in the spaces above, and also at the top of every page.

Problem	Possible Points	Points Earned
1	18	
2	15	
3	12	
4	12	
5	20	
6	13	
7	10	
PreTotal	100	
Extra Credit	8	
PostTotal	108	

CS 421 Midterm 2 **Name:**_____

1. (18 points) Give a most general unifier for the following unification problem. Capital letters **A** and **B** denote variables of unification and lower case letters **f, p** and **s** denote constructors. Show all your work by listing the operation being performed at each step, and all the results of that operation, including intermediate substitutions generated. Your final result must be expressed as a single simultaneous substitution, not a composition of individual substitutions.

{(f(A,C) = B); (p(s(A), s(B)) = p(C, s(B)))}

Note: The notation f o g will refer to the function that first does g and then applies f to the result. f o g (x) = f (g (x)),.

Solution:

Unify {(f(A,C) = B); (p(s(A), s(B)) = p(C, s(B)))}
= Unify {(B = f(A,C)); (p(s(A), s(B)) = p(C, s(B)))} by Orient (f(A,C) = B)
= Unify {(p(s(A), s(f(A,C))) = p(C, s(f(A,C)))} with {B → f(A,C)}
by Eliminate (B = f(A,C))
= Unify {(s(A) = C); (s(f(A,C)) = s(f(A,C)))} with {B → f(A,C)}
by Decompose (p(s(A), s(f(A,C))) = p(C, s(f(A,C)))
= Unify {(C = s(A)); (s(f(A,C)) = s(f(A,C)))} with {B → f(A,C)} by Orient (s(A) → C)
= Unify {(s(f(A, s(A))) = s(f(A, s(A))))} with {C → s(A); B → f(A, s(A))}
by Eliminate (C = s(A))
= Unify {} with {C → s(A); B → f(A, s(A))} by Delete
= {C → s(A); B → f(A, s(A))}

CS 421 Midterm 2 **Name:**_____

2. (12 pts total) In parts a) through c) you are to give a regular expression generating each of the following languages over the alphabet $\Sigma = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$. You should use the notation given in class: Regular expressions over an alphabet Σ are strings over Σ together with the five extra characters **(**, **)**, *****, **∨**, and **ε**. **No other symbols should occur in your regular expression, and they will not be accepted**. In part d) you are asked to give a regular grammar for the language in a).

   a. (2 pts) Give a regular expression for the set of all strings that either contain no **a**'s or no **b**'s.

   Soultion:
   (b ∨ c)* ∨ ((a ∨ b)*)

   b. (4 pts) Give a regular expression for the set of all strings such that no **a** may be followed at any distance by a **b**, no **b** may followed at any distance by a **c**, and no **c** may be followed at any distance by an **a**.

   Solution:
   (a*c*) ∨ (b*a*) ∨ (c*b*)

CS 421 Midterm 2 **Name:**_____

2. (12 pts total) In parts a) through c) you are to give a regular expression generating each of the following languages over the alphabet Σ = {**a**, **b, c**}. You should use the notation given in class: Regular expressions over an alphabet Σ are strings over Σ together with the five extra characters **(**, **)**, *****, **∨**, and **ε**. **No other symbols should occur in your regular expression, and they will not be accepted**. In part d) you are asked to give a regular grammar for the language in a).

c. (4 pts) Give a regular expression for the set of all strings that do not end in **cab**.

Solution:
ε ∨ ((a ∨ b ∨ c)*(a ∨ c)) ∨ ((a ∨ b ∨ c)*(b ∨ c)b) ∨ ((a ∨ b ∨ c)*(a ∨ b)ab)

d. (3 pts) Give a regular grammar for the set of all strings that either contain no **a**'s or no **b**'s.

Solution:
S ::= a A | b B | c S | ε
A ::= b A | c A | ε
B ::= a B | c B | ε

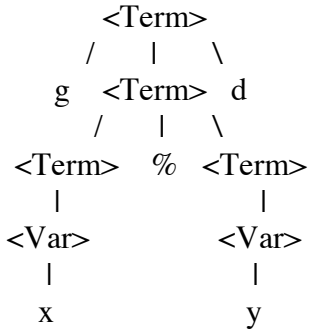
CS 421 Midterm 2 **Name:**_____

3. (12 points total) Given the following BNF grammar, for each of the following strings, give a parse tree for it, if it parses starting with **<Term>,** or write "None exists" if it does not parse starting with **<Term>.** The terminals for this grammar are **{%, b, g, d, x, y, z}**.

<Term> ::= <Var> | <Term> % <Term> | <Factor> b | g <Term> d
<Factor> ::= g <Term> | g <Factor>
<Var> ::= x | y | z

a. (2 pts) **g x % y d**

Solution:



CS 421 Midterm 2 **Name:**_____

3. (cont) Given the following BNF grammar, for each of the following strings, give a parse tree for it, if it parses starting with **<Term>,** or write "None exists" if it does not parse starting with **<Term>.** The terminals for this grammar are **{%, b, g, d, x, y, z}**.

<Term> ::= <Var> | <Term> % <Term> | <Factor> b | g <Term> d
<Factor> ::= g <Term> | g <Factor>
<Var> ::= x | y | z

b. (5 pts) **g x % g g y b**

Solution: None exists

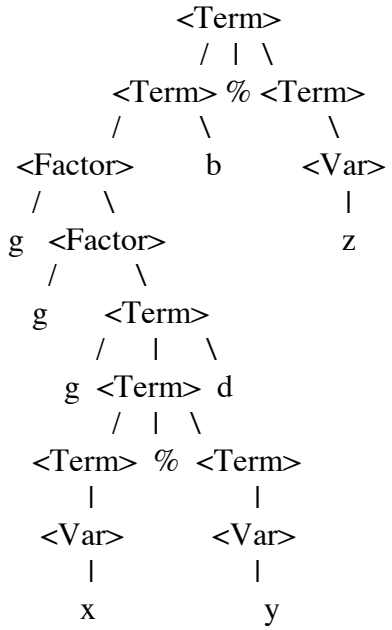
CS 421 Midterm 2 **Name:**_____

3. (cont) Given the following BNF grammar, for each of the following strings, give a parse tree for it, if it parses starting with **<Term>,** or write "None exists" if it does not parse starting with **<Term>.** The terminals for this grammar are **{%, b, g, d, x, y, z}**.

<Term> ::= <Var> | <Term> % <Term> | <Factor> b | g <Term> d
<Factor> ::= g <Term> | g <Factor>
<Var> ::= x | y | z

c. (5 pts) **g g g x % y d b % z**

Solution:



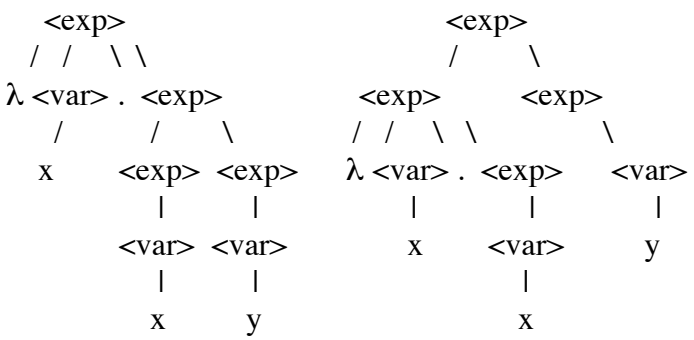
CS 421 Midterm 2 **Name:**_____

4. (18 points total) Consider the following grammar over the alphabet {**λ, . , (,), x, y, z**}:

<exp> ::= <var> | λ <var> . <exp> | <exp> <exp> | (<exp>)
<var> ::= x | y | z

a. (6 pts) Show that the above grammar is ambiguous (using the definition of an ambiguous grammar).

Solution: λ x . x y has two parses



CS 421 Midterm 2 **Name:**_____

4. (18 points total) Consider the following grammar over the alphabet {**λ, . , (,), x, y, z**}:

<exp> ::= <var> | λ <var> . <exp> | <exp> <exp> | (<exp>)
<var> ::= x | y | z

b. (12 pts) Write a new grammar accepting the same language accepted by **<exp>** above, and such that application (**<exp> <exp>**) associates to the left and has higher precedence than abstraction (**λ <var> . <exp>**).

Solution:

<exp> ::= <no_abs> <no_app> | <no_app>
<no_abs> ::= <no_abs> <atom> | <atom>
<no_app> ::= λ <var> . <exp> | <atom>
<atom> ::= <var> | (<exp>)
<var> ::= x | y | z

CS 421 Midterm 2
Name:_____

5. (20 points total) Consider the following grammar:

<expr> ::= <atom> | <atom> * <expr>
<atom> ::= 0 | (<expr>)

a. (3 pts) Write an Ocaml data type **token** for the tokens that lexer would generate as input to a parser for this grammar.

Solution:

```
type token = Star | Zero | LPar | RPar
```

b. (4 pts) Write Ocaml data types **expr** and **atom** to represent parse trees for each of the syntactic categories in the given grammar.

Solution:

```
type expr = AtomExpr of atom | StarExpr of (atom * expr)
and atom = ZeroAtom | ParensAtom of expr
```

CS 421 Midterm 2 **Name:**_____

5. (cont)(20 points total) Consider the following grammar:

<expr> ::= <atom> | <atom> * <expr>
<atom> ::= 0 | (<expr>)

c. (13 pts)Using the types you gave in parts **a)** and **b)**, write an Ocaml recursive descent parser **parse: token list -> expr** that, given a list of **token**s, returns an **expr** representing an **<expr>** parse tree. You should use

raise (Failure "no parse")

for cases where no parse exists.

Solution:

```
let rec expr tokens =
   match atom tokens with (atom_parse, s_toks) ->
    (match s_toks with Star::e_toks ->
      (match expr e_toks
       with (expr_parse, toks) -> (StarExpr(atom_parse, expr_parse),toks))
     | _ -> (AtomExpr atom_parse, s_toks))
and atom tokens =
    match tokens
     with Zero::toks1 -> (ZeroAtom, toks1)
    | LPar :: toks1->
     (match expr toks1 with (expr_parse, toks2) ->
      (match toks2 with RPar::toks3 -> (ParensAtom expr_parse, toks3)
        | _ -> raise (Failure "no_parse")))
    | _ -> raise (Failure "no_parse")

let parse tokens = match expr tokens with (e,[]) -> e | _ -> raise (Failure "no_parse")
```

CS 421 Midterm 2
Workspace

Name:_____

CS 421 Midterm 2 **Name:**_____

6. (13 pts) Given the following grammar over nonterminal **<b>** and **<e>**, and terminals **z, l, r, p** and **eof**, with start symbol **<b>**:

   P0: **<b> ::= <e> eof**
   P1: **<e> ::= z**
   P2: **<e>::= l <e> r**
   P3: **<e>::= <e> p**

   and Action and Goto tables generated by YACC for the above grammar:

State	Action						Goto	
	z	**l**	**r**	**p**	**[eof]**		**<b>**	**<e>**
st1	**s 3**	**s 4**	**err**	**err**	**err**			**st5**
st2	**err**	**err**	**err**	**err**	**a**			
st3	**r 1**	**r 1**	**r 1**	**r 1**	**r 1**			
st4	**s 3**	**s 4**	**err**	**err**	**err**			**st6**
st5	**err**	**err**	**err**	**s 7**	**a**			
st6	**err**	**err**	**s 8**	**s 7**	**err**			
st7	**r 3**	**r 3**	**r 3**	**r 3**	**r 3**			
st8	**r 2**	**r 2**	**r 2**	**r 2**	**r 2**			

   where **st*i*** is state *i*, **s *i*** means **shift *i***, **r *i*** means **reduce *i*,** **a** means **accept** and [**eof]** means we have reached the end of input, describe how the string **lzprp[eof]** would be parsed with an LR parser using these productions and tables by filling in the table on the next page. I have given you the first 5 cells to get started. **Caution:** There are strictly more rows than you will need, so do not expect to fill them all.

CS 421 Midterm 2 **Name:**_____

Stack	**Current String**	**Action**
Empty	**lzprp[eof]**	Initialize stack, go to state 1
st1	**lzprp[eof]**	Shift, go to state 4
st1:l:st4	zprp[eof]	Shift, go to state 3
st1:l:st4:z:st3	prp[eof]	Reduce by P1, go to state 6
st1:l:st4:<e>:st6	prp[eof]	Shift, go to state 7
st1:l:st4:<e>:st6:p:st7	rp[eof]	Reduce by P3, go to state 6
st1:l:st4:<e>:st6	rp[eof]	Shift, go to state 8
st1:l:st4:<e>:st6:r:st8	p[eof]	Reduce by P2, go to state 5
st1:<e>:st5	p[eof]	Shift, go to state 7
st1:<e>:st5:p:st7	[eof]	Reduce by P3, go to state 5
st1:<e>:st5	[eof]	accept

CS 421 Midterm 2 **Name:**_____

7. (16 points total)
   a. (6 pts) Give a natural semantics (a.k.a. structured operational semantics) derivation of the evaluation of:

$$((\mathbf{x:= 3; x := x + 2}), \{\mathbf{x} \rightarrow \mathbf{1}\})$$

.

$$\dfrac{\dfrac{}{(\mathrm{x:= 3}, \{\mathbf{x} \rightarrow \mathbf{1}\}) \Downarrow \{\mathbf{x} \rightarrow \mathbf{3}\}} \quad \dfrac{\dfrac{}{(\mathrm{x}, \{\mathbf{x} \rightarrow \mathbf{3}\}) \Downarrow 3} \quad \dfrac{}{(2, \{\mathbf{x} \rightarrow \mathbf{3}\}) \Downarrow 2} \quad 3+2=5}{(\mathbf{x := x + 2}, \{\mathbf{x} \rightarrow \mathbf{3}\}) \Downarrow \{\mathbf{x} \rightarrow \mathbf{5}\}}}{((\mathbf{x:= 3; x := x + 2}), \{\mathbf{x} \rightarrow \mathbf{1}\}) \Downarrow \{\mathbf{x} \rightarrow \mathbf{5}\}}$$

CS 421 Midterm 2 **Name:**_____

a. (16 points total) (cont)

b. (10 pts) Give a transition semantics derivation of the evaluation of:

((**x:= 3; x := x + 2**), {**x → 1**})

$$\dfrac{\dfrac{}{(x := 3, \{x \to 1\}) \to \{x \to 3\}}}{((x:= 3; x := x + 2), \{x \to 1\}) \to (x := x + 2, \{x \to 3\})}$$

$$\dfrac{\dfrac{\dfrac{}{(x, \{x \to 3\}) \to (3, \{x \to 3\})}}{(x + 2, \{x \to 3\}) \to (3 + 2, \{x \to 3\})}}{(x := x + 2, \{x \to 3\}) \to (x := 3 + 2, \{x \to 3\})}$$

$$\dfrac{\dfrac{}{(3 + 2, \{x \to 3\}) \to (5, \{x \to 3\})}}{(x := 3 + 2, \{x \to 3\}) \to (x := 5, \{x \to 3\})}$$

$$\dfrac{}{(x := 5, \{x \to 3\}) \to \{x \to 5\}}$$

CS 421 Midterm 2 **Name:**_____

b. (Extra Credit) (8 points total) Assuming you have written a lexer for a programming language without comments with entry point called **token** (taking no extra arguments), what code would you need to add to the header, the body of **token** and extra entry points to add comments that start with **/*** and end with */ and allow for comments to be nested within comments.
Do not assume that we are track line and character numbers. You should
raise (Failure "No closing comment")
if the end of the file is reached after a /* and before a matching */.

a. Changes to header (if any):
Soultion:

```
let open_comment = "/*"
let close_comment = "*/"
```

b. Changes to the body of **token** (if any):

Solution:

```
| open_comment          {comment 1 lexbuf}
```

c. Additional entry points (if any):

Solution:

```
and comment lpars = parse
    open_comment     {comment (lpars + 1) lexbuf}
  | close_comment   { if lpars = 1 then token lexbuf else comment (lpars – 1) lexbuf }
  |  eof           { raise (Failure "No closing comment") }
  | _  { comment lpars lexbuf }
```

CS 421 Midterm 2 **Name:**_____

Simple Imperative Programming Language

$I \in$ *Identifiers* $N \in$ *Numerals*

B ::= true | false | B & B | B or B | not B | $E < E$ | $E = E$

$E ::= N \mid I \mid E + E \mid E * E \mid E - E \mid - E$

C::= skip | C;C | I ::= E | if B then C else C fi | while B do C od

Natural Semantics Rules

Identifiers: $(I,m) \Downarrow m(I)$ **Numerals are values:** $(N,m) \Downarrow N$

Booleans: (true,m) $\Downarrow$ true (false ,m) $\Downarrow$ false

$$\frac{(B, m) \Downarrow \text{false}}{(B \,\&\, B', m) \Downarrow \text{false}} \qquad \frac{(B, m) \Downarrow \text{true} \quad (B', m) \Downarrow b}{(B \,\&\, B', m) \Downarrow b}$$

$$\frac{(B, m) \Downarrow \text{true}}{(B \text{ or } B', m) \Downarrow \text{true}} \qquad \frac{(B, m) \Downarrow \text{false} \quad (B', m) \Downarrow b}{(B \text{ or } B', m) \Downarrow b}$$

$$\frac{(B, m) \Downarrow \text{true}}{(\text{not } B, m) \Downarrow \text{false}} \qquad \frac{(B, m) \Downarrow \text{false}}{(\text{not } B, m) \Downarrow \text{true}} \qquad \frac{(E, m) \Downarrow U \quad (E', m) \Downarrow V \quad U \sim V = b}{(E \sim E', m) \Downarrow b} \quad (\sim \text{ a relation})$$

Arithmetic Expressions: $\dfrac{(E, m) \Downarrow U \quad (E', m) \Downarrow V \quad U \; op \; V = N}{(E \; op \; E', m) \Downarrow N}$
(*op* an arith binary operation)

Commands:

Skip: (skip, m) $\Downarrow m$ **Assignment:** $\dfrac{(E,m) \Downarrow V}{(I ::= E, m) \Downarrow m[I \leftarrow V]}$

Sequencing: $\dfrac{(C,m) \Downarrow m' \quad (C',m') \Downarrow m''}{(C;C', m) \Downarrow m''}$

If Then Else Command:

$$\frac{(B,m) \Downarrow \text{true} \quad (C,m) \Downarrow m'}{(\text{if } B \text{ then } C \text{ else } C' \text{ fi}, m) \Downarrow m'} \qquad \frac{(B,m) \Downarrow \text{false} \quad (C',m) \Downarrow m'}{(\text{if } B \text{ then } C \text{ else } C' \text{ fi}, m) \Downarrow m'}$$

While Command:

$$\frac{(B,m) \Downarrow \text{false}}{(\text{while } B \text{ do } C \text{ od}, m) \Downarrow m} \qquad \frac{(B,m) \Downarrow \text{true} \quad (C,m) \Downarrow m' \quad (\text{while } B \text{ do } C \text{ od}, m') \Downarrow m''}{(\text{while } B \text{ do C od}, m) \Downarrow m''}$$

CS 421 Midterm 2 **Name:**_____

Transition Semantics:

Identifiers: $(I,m) \rightarrow m(I)$ **Numerals are values:** $(N,m) \rightarrow N$

Booleans:

(false & B, m) → (false, m) (true & B, m) → (B,m) $\dfrac{(B, m) \rightarrow (B'', m)}{(B \,\&\, B', m) \rightarrow (B'' \,\&\, B', m)}$

(true or B, m) → (true, m) (false or B, m) → (B,m) $\dfrac{(B, m) \rightarrow (B'', m)}{(B \text{ or } B', m) \rightarrow (B'' \text{ or } B', m)}$

(not true, m) → (false, m) (not false, m) → (true, m) $\dfrac{(B, m) \rightarrow (B', m)}{(\text{not } B, m) \rightarrow (\text{not } B', m)}$

$\dfrac{(E, m) \rightarrow (E'', m)}{(E \sim E', m) \rightarrow (E'' \sim E', m)}$ $\dfrac{(E, m) \rightarrow (E', m)}{(V \sim E, m) \rightarrow (V \sim E', m)}$ ~ a relation

($U \sim V$, m) → true or false, depending on whether $U \sim V$ holds or not

Arithmetic Expressions:

$\dfrac{(E, m) \rightarrow (E'', m)}{(E \text{ op } E', m) \rightarrow (E'' \text{ op } E', m)}$ $\dfrac{(E, m) \rightarrow (E', m)}{(V \text{ op } E, m) \rightarrow (V \text{ op } E', m)}$

(U op V, m) → (N,m) where $N = U$ op V

Commands:

(skip, m) → m $\dfrac{(E,m) \rightarrow (E',m)}{(I ::= E,m) \dashrightarrow (I ::= E',m)}$ (I::=V,m) → $m[I \leftarrow V]$

$\dfrac{(C,m) \rightarrow (C'',m')}{(C;C', m) \rightarrow (C'';C', m')}$ $\dfrac{(C,m) \rightarrow m'}{(C;C', m) \rightarrow (C', m')}$

If Then Else Command:

(if true then C else C' fi, m) → (C, m) (if false then C else C' fi, m) → (C', m)

$\dfrac{(B,m) \rightarrow (B',m)}{(\text{if } B \text{ then } C \text{ else } C' \text{ fi}, m) \rightarrow (\text{if } B' \text{ then } C \text{ else } C' \text{ fi}, m)}$

While Command:

(while B do C od, m) → (if B then C; while B do C od else skip fi, m)