THE UNIVERSITY OF CALGARY

FACULTY OF SCIENCE

FINAL EXAMINATION

COMPUTER SCIENCE 411

April, 2014 Time: 2 hrs.

Instructions

The exam contains questions worth 100 points. Answer all questions.

This exam is closed book. You are expected to *explain* your answers.

. 2

30 marks

1. Consider the following grammar:

```
a -> a b A B
   | a B
   | C.
b -> a b X Y
   | b Y
   |.
```

   (a) Calculate "vital statistics" of the grammar, that is the *first* and *follow* sets. Indicate which non-terminals are nullable and which are endable.

   (b) Give the LR(0)-item automaton: is the grammar SLR(1) or LALR(1)?

   (c) What must be satisfied by a grammar to ensure it is LL(1)? Is this grammar LL(1)?

   (d) Can you transform the grammar to remove left recursion? Is this transformed grammar LL(1)?

   (e) Is this grammar ambiguous?

20 marks

2. (a) Draw a Venn diagram to show the relationship between the grammar classes: LR(0), LL(0), LALR(1), SLR(1), LR(1), and LL(1). Are there any LL(0) grammars which are not LR(0); are there any LL(1) grammars which are not LALR(1)?

   (b) Draw the LR(0)-item automaton for the following grammar:

```
e -> ID [ e t ]
     | .
t -> t ( t e )
     | .
```

   Indicate all shift/reduce and reduce/reduce conflict in the LR(0) automaton. Is the grammar LR(0)? Is it SLR(1)? Is it LALR(1)?

25 marks

3. This question is concerned with the organization of stack-based runtime environment for the **M** language:

   (a) Describe the organization of an *activation record* for an **M** function.

   (b) Explain the purpose of the *return pointer* and the *dynamic link* in a function activation record.

   (c) Explain what is meant by the "static distance" of an occurrence of an identifier from its place of declaration in a program. Explain how this information is used to access the value of a *non-local* variables.

   (d) Explain what happens when a simple **M** function is called (no arrays or data).

   (e) Explain where and how the space for dynamic (multi-dimensional) arrays is allocated in **M**. Explain how arrays are passed to functions and how the function knows how many dimensions the array has and the size of each dimension.

   (f) Where is the space for user-defined data types allocated? How is a case statement for a datatype implemented?

25 marks

4. Given the following abstract syntax tree for "where expressions" with simultaneous declarations draw the plumbing diagrams and develop pseudo-code to determine the legality and type of such an expression.

   The definition of the abstract syntax tree for "where expressions" is:

```
data Exp = ADD Exp Exp | MUL Exp Exp
         | AND Exp Exp | OR Exp Exp
         | LT Exp Expr | EQ Exp Exp
         | ID String | VI Int | VB Bool
         | WHERE Exp [(String,Exp)]
```

   A typical syntax tree (which does type) is:

```
AND (WHERE (ID x) [(x,VB True)])
    (WHERE (LE (ID y) (ID x)) [(y,VI 27)
                              ,(x,VI 54)])
```

   which may be viewed as representing the program:

```
(x where x = true)
&& (y < x where y = 27; x = 54)
```

   The operations should be assumed to have the following types:

```
OR,AND::(BOOL,BOOL)->BOOL,
ADD,MUL::(INT,INT)->INT,
LT, EQ::(INT,INT)->BOOL
```

   In a given `where` declaration list only one declaration for each symbol is allowed. As the declarations are simultaneous, the body of each declaration can only use variables which have already been defined in an outer scope.

You may assume you have an implementation of a simple symbol table, of type `ST`, which allows `look_up` of variables (returning their most recent type), and `insert`ion of types of variables:

```
data Found_or_not = FOUND Type
                  | NOT_FOUND
data Type = INT | BOOL

look_up:: ST -> String -> Found_or_not
insert:: ST -> (String,Type) -> ST
```