UNIVERSITY OF DUBLIN

TRINITY COLLEGE

Faculty of Engineering, Mathematics & Science
School of Computer Science & Statistics

Integrated Computer Science
B.A. (Mod.) CSLL

Trinity Term 2013

Compiler Design (CS3071)

Saturday 11th May 2013 Gold Hall (220) 09:30 – 11:30

Dr DM Abrahamson

Instructions to Candidates

Answer question number 1 and one other question

Materials permitted for this examination

None

1. Consider the following context free grammar for arithmetic expressions with starting non-terminal symbol <E>:

```
<E> → <E> + <T>
<E> → <T>
<T> → <T> * <P>
<T> → <P>
<P> → ( <E> )
<P> → IDENT
<P> → CONST
```

where the lexical token IDENT represents a variable identifier of type INT or FLOAT, and CONST represents a numeric constant.

i. Extend the grammar to include subtraction, division and exponentiation (where exponentiation is right associative and has higher precedence than multiplication and division). [6 Marks]

ii. Produce an equivalent LL(1) grammar by left-factoring and by eliminating left recursion from the productions. [6 Marks]

iii. Compute the selection set for each of the productions in the LL(1) grammar. [8 Marks]

iv. Add attributes and attribute evaluation rules to the productions for the grammars obtained in parts i and ii above so that each of their starting non-terminal symbols will have a single synthesized attribute whose value is that of the expression being parsed. [12 Marks]

v. Show that both grammars produce the same translation by drawing fully decorated derivation-trees for the expression 2↑3↑2+A/B where variables A and B have values 256 and 32 respectively, and ↑ is the (right-associative) exponentiation operator. [8 Marks]

vi. Rewrite the productions from the LL(1) grammar in simple assignment form, and design corresponding stack replacement operations for an augmented pushdown machine. [12 Marks]

vii. Show the contents of the augmented pushdown machine's stack at each step along the way while parsing the expression (A-B)*C. [8 Marks]

[Total 60 Marks]

2. Design an L-attributed translation grammar production for a basic loop statement of the form:

   ```
   for i := 1 to n do <stmts...>
   ```

   where the control variable *i* is assigned successive values running from *1* to *n*. Describe the function of the action symbols and the information represented by the attributes. Care should be taken to ensure that the generated code will run correctly for all possible values of *n*. [40 Marks]

3. Design L-attributed translation grammar productions to process constant definitions and variable declarations of the following form:

   ```
   CONST
      max = 512;
      marker = "$";
   VAR
      ch: CHAR;
      top: INTEGER;
      stack: ARRAY [1..max] OF CHAR;
   ```

   Describe the function of the action symbols and the information represented by the attributes, explain the mapping of arrays and show by example the structure and contents of the symbol-table. [40 Marks]