TRINITY COLLEGE DUBLIN
THE UNIVERSITY OF DUBLIN

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

Integrated Computer Science
B.A. (Mod.) CSLL
Year 3 Annual Examinations

Trinity Term 2015

Compiler Design (CS3071)

Wednesday 13th May 2015 Luce Lower 14:00 – 16:00

Dr DM Abrahamson

Instructions to Candidates

Answer question number 1 and one other question

Materials permitted for this examination

None

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

```
<E>  →  <E> + <T>
<E>  →  <T>
<T>  →  <T> * <P>
<T>  →  <P>
<P>  →  ( <E> )
<P>  →  const
```

where the lexical token `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 subtraction, etc).

[6 Marks]

ii. Produce an equivalent LL(1) grammar for the language by left-factoring and by eliminating left recursion from the productions obtained in part i above.

[6 Marks]

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

[8 Marks]

iv. Design an interpreter for arithmetic expressions by adding attributes and attribute evaluation rules to the productions for the LL(1) grammar obtained in part ii above so that the starting non-terminal symbol will have a single synthesized attribute whose value is that of the expression being parsed.

[8 Marks]

v. Using productions from the grammar obtained in part iv above, draw a fully decorated derivation-tree for the expression `2↑3↑2+A/B-C` where `A`, `B` and `C` are numeric constants with values `256`, `32` and `16` respectively, and ↑ represents the (right-associative) exponentiation operator.

[8 Marks]

vi. Rewrite the attributed LL(1) productions obtained in part iv above in simple assignment form, and design corresponding stack replacement operations for an augmented pushdown machine.

[12 Marks]

vii. Using the stack replacements obtained in part vi above, show the contents of the augmented pushdown machine's stack at each step along the way while parsing

2. Describe the information that should be maintained in the symbol table at compile time to record the properties of multi (eg one and two) dimensional arrays, and design L-attributed translation grammar productions to parse variable declarations of the general form:

```
VAR
   data: ARRAY [1..16] OF INTEGER;
   table: ARRAY [1..8, 1..32] of STRING;
```

Explain in detail the function of the action symbols, and sketch the symbol table's contents when compiling the variable declarations shown above.

[40 Marks]

3. Discuss the relationship between the output action symbols {label_p} and {$\text{jumpf}_{p,q}$}, and describe in detail their use in L-attributed translation grammar productions for `IF` and `REPEAT` statements of the form:

```
<statement> → IF <condition> THEN <statement>
<statement> → REPEAT <statements> UNTIL <condition>
```

Explain the role of the function "`newl`" and demonstrate, by example, how the processing of the address field in a generated branch instruction is dependent on the relative position of a {label} and its corresponding {jumpf} action.

[40 Marks]