**UNIVERSITY OF LONDON**

**GOLDSMITHS COLLEGE**

**B.Sc. Examination 2012**

## COMPUTING AND INFORMATION SYSTEMS

# IS53011A Language Design and Implementation

Duration: 2 hours 15 minutes

Date and time:

---

*There are five questions on this paper. You should answer no more that THREE questions. Full marks will be awarded for complete answers to a total of THREE questions. Each question carries 25 marks. The marks for each part of a question are indicated at the end of the part in [.] brackets.*

*There are 75 marks available on this paper.*

*No calculators should be used.*

**THIS PAPER MUST NOT BE REMOVED FROM THE EXAMINATION ROOM**

**Question 1.**

a) Plot a picture of the algorithmic structure of the back end of a compiler. **[4]**

b) Define what is meant by one operator having a higher preference than another operator in a programming language. **[3]**

c) i) Explain briefly when we use regular grammars and when we use context-free grammars in compiler design. **[5]**

ii) Give five initial strings that can be generated using the regular grammar: $a\ (\ b\ |\ a{*}c\ ){*}$. **[4]**

iii) Construct a regular expression that defines binary numbers that are multiples of two. **[3]**

d) Let the following grammar for expressions with balanced parentheses be given:

$$E \rightarrow EE \mid (\ E\ ) \mid \in$$

Check if this grammar is ambiguous or unambiguous by developing a parse tree for the following simple string: ()()(). **[6]**

**Question 2.**

a) Define recursively the notion of regular expressions over a given alphabet with elements: characters { *a*, *b* }, empty string {∈}, and relevant operations on them. **[5]**

b) Describe the nature of strings produced by the following grammar, in terms of the ordering of *a*'s and *b*'s:
$b^*\ (\ abb^*\ )^*\ (\ a \mid \in\ )$. **[3]**

c) i) Design a nondeterministic finite state automaton (NFA) using the Thompson's construction algorithm for the following regular expression: $a\ (\ abc \mid c\ )^*$. **[5]**

ii) Convert the designed NFA into a deterministic finite-state automaton (DFA) using the subset construction algorithm, showing the operations of the ∈-*closure* and *move* functions. **[9]**

iii) Develop the transition table for the generated DFA from part ii). **[3]**

**Question 3.**

a) Which are the main four components of a context-free grammar? **[2]**

b) Eliminate the immediate left recursion from the following context-free grammar: **[4]**

$S \rightarrow SB \mid aE$

$B \rightarrow bE \mid d$

$E \rightarrow Ea \mid c$

c) You are given the following context-free grammar, which is suitable for top-down parsing:

$S \rightarrow aTFa$

$T \rightarrow bTb \mid a$

$F \rightarrow cFc \mid a$

Demonstrate the performance of the nonrecursive parser on the input string: $ababcaca\,\$$ using the following parsing table:

| | $a$ | $b$ | $c$ | $\$$ |
|---|---|---|---|---|
| $S$ | $S \rightarrow aTFa$ | | | |
| $T$ | $T \rightarrow a$ | $T \rightarrow bTb$ | | |
| $F$ | $F \rightarrow a$ | | $F \rightarrow cFc$ | |

i) Show the stack, the input and the output of the nonrecursive parser at each step. **[14]**

ii) Write down the leftmost derivation of the given input string: $ababcaca\,\$$ according to the output of the parser. **[5]**

**Question 4.**

a) Explain briefly the steps of the closure operation for developing parsing tables for bottom-up shift-reduce parsing. **[5]**

b) You are given the following grammar, which is suitable for bottom-up parsing:

(1) $S' \rightarrow S$

(2) $S \rightarrow T{=}F$

(3) $F \rightarrow T$

(4) $T \rightarrow x$

(5) $T \rightarrow x{+}x$

Consider the following parsing table:

| | Action | | | | Goto | | |
|---|---|---|---|---|---|---|---|
| State | *x* | + | = | $ | S | T | *F* |
| 0 | *s3* | | | | *1* | *2* | |
| 1 | | | | *accept* | | | |
| 2 | | | *s4* | *r3* | | | |
| 3 | | *s6* | *r4* | *r4* | | | |
| 4 | *s3* | | | | | *8* | *5* |
| 5 | | | | *r2* | | | |
| 6 | *s7* | | *r4* | *r4* | | | |
| 7 | | | *r5* | *r5* | | | |
| 8 | | | | *r3* | | | |

i) Develop the canonical collection of items from this grammar using the sets-of-items construction algorithm. **[8]**

ii) Demonstrate the moves of the bottom-up shift-reduce parser on the input string: $x{=}x{+}x\ \$$ by showing the stack, the input and the output. **[10]**

iii) Draw the parse tree produced by the parser. **[2]**

**Question 5.**

a) Give the two most important properties that an optimising compiler should provide. **[4]**

b) Consider the following implementation of the sort function:

```
void Sort( int a[] )
{
    int i, j, x;
     i = 1;
     while ( i < 5 )
     {
         j = i;
         while ( j > 0 )
         {
              x = a[ j-1 ];
              a[ j-1 ] = a[ j ];
              a[ j ] = x;
              --j;
         }
     }
     ++i;
}
```

i) Translate this function into a three-address intermediate code. **[10]**

ii) Optimise the developed three-address code by elimination of the induction variables in the loops, and also eliminate the dead code. **[8]**

iii) Where does the name "three-address code" in the field of computer programming language design comes from? **[3]**