EXAMINATION PAPER PROFORMA

This form should be attached to every examination paper submitted to the Head of Assessments.

Name of Unit / Element: Language Design and Implementation..........................

Code Number:.......................... IS53011A ....

Number of Pages:......................6..

May the paper be backed? No
(Delete as appropriate)

May the paper be reduced? No
(Delete as appropriate)

Length of paper: 2 hours 15 minutes
(Complete or delete as appropriate)

Number of students expected to sit paper: ....

Are the students permitted to retain the paper? No
(Delete as appropriate)

Name of member of staff responsible for paper: ....Dr N. Nikolaev..............

Any other special requirements: ....

....

....

I confirm that the above paper has been correctly scrutinised and agreed by the External Examiner. Also that, if applicable, the necessary copyright permission has been granted.

Signed:... Date: ...... 4 April 2011.................................

Name: ..... N. NIKOLAEV .... Department: ......Computing............................
(Block capitals)

C:\GMDH\Nik\Exms\ExaminationPaperProforma-1.doc
MW

UNIVERSITY OF LONDON

GOLDSMITHS COLLEGE

B.Sc. Examination 2011

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.

THIS PAPER MUST NOT BE REMOVED FROM THE EXAMINATION ROOM

Question 1.

a) What are the main 6 phases of a programming language compiler? **[3]**

b) Given the following programming language grammar:

$$S \rightarrow ES \mid b$$
$$E \rightarrow SE \mid a$$

Construct parse trees for the expression: *abbab,* in order to find out if this grammar is ambiguous. **[6]**

c) Find a rightmost derivation of the string: *aabbaa*, using the following grammar: **[4]**

$$S \rightarrow aES \mid a$$
$$E \rightarrow ba \mid SbE \mid SS$$

d) Develop a language grammar that generates binary numbers using *0* and *1*. **[4]**

e) Eliminate the left recursion from the following grammar: **[8]**

$$E \rightarrow a \mid [\ S\ ]$$
$$S \rightarrow b \mid S\,;E$$

Question 2.

a) Represent the regular expression: $a \mid bc^*$ by a corresponding context-free grammar. **[4]**

b) Give a definition of the notion of deterministic finite-state automaton (DFA). **[3]**

c) Consider the following regular expression: $b^* \ (\ a \mid b \ )^*$.

i) Design a nondeterministic finite state automaton (NFA) for this regular expression using Thompson's construction algorithm. **[5]**

ii) Transform this NFA into a corresponding deterministic finite-state automaton (DFA) using the subset construction algorithm. Demonstrate the $\in$-*closure* and *move* functions. **[9]**

iii) Draw the derived DFA as a graph and identify the accepting states. **[4]**

Question 3.

a) What is the aim of the recursive-descent parser, and what does it produce as a result? **[3]**

b) Draw a model of a nonrecursive predictive parser and name each component in it. **[4]**

c) Consider the following context-free grammar for parsing:

$$E \rightarrow aE \mid d$$
$$E \rightarrow bSc$$
$$S \rightarrow ET$$
$$T \rightarrow ;S \mid \epsilon$$

i) Using the parsing table given below, simulate the performance of the nonrecursive predictive parsing algorithm on the input string: *bad;dc* \$. Show the stack, the input and the output of the nonrecursive parser at each algorithmic step. **[14x1=14]**

	a	b	c	d	$;$	\$
E	$E \rightarrow aE$	$E \rightarrow bSc$		$E \rightarrow d$		
S	$S \rightarrow ET$	$S \rightarrow ET$		$S \rightarrow ET$		
T			$T \rightarrow \epsilon$		$T \rightarrow ;S$	

ii) Draw the derived parse tree for the given input string: *bad;dc* \$. **[4]**

Question 4.

a) i) What is the purpose of the bottom-up shift-reduce parser? **[3]**

ii) Explain the abbreviation of LR(1) grammars. **[2]**

iii) For which class of programming languages can we constructs LR parsers? **[2]**

b) Consider the following grammar suitable for bottom-up parsing:

(1) $S' \rightarrow S$

(2) $S \rightarrow S+E$

(3) $S \rightarrow a$

(4) $E \rightarrow b$

i) Compute the *FOLLOW* functions for the nonterminals in this grammar. **[2]**

ii) Construct the canonical collection of items from this grammar using the sets-of-items construction algorithm. **[5]**

iii) Interpret the performance of the bottom-up shift-reduce parser on the input string: $a+b+b$ \$ using the parsing table given below. Demonstrate the stack, the input and the output. **[11]**

	Action				Goto	
State	*a*	*b*	+	\$	*S*	*E*
0	*s1*				2	
1			*r3*	*r3*		
2			*s3*	*accept*		
3		*s4*				5
4			*r4*	*r4*		
5			*r2*	*r2*		

Question 5.

a) i) Which are the main code improving transformations used in optimising compilers? **[3]**

ii) Which code improving transformation is called local and which global? **[4]**

b) Given the following implementation of the binary search algorithm:

```
void BinarySearch( int x[], int N, int s )
{
    int v, z, y, l, r, N;
    l = 1; r = N; z = -1;
    while ( r >= l )
    {
        v = (int)( l + r ) / 2;
        if ( s < x[ v ] )  r = v - 1; else l = v + 1;
        if ( s == x[ v ] ) { z = 1; break; }
    }
    y = z; print( "result = ", y );
}
```

i) Generate three-address intermediate code for this binary search algorithm. **[10]**

ii) Transform the generated three-address code by elimination of the common subexpressions to avoid recomputations, next eliminate the dead code, and finally rewrite the whole code for the algorithm making the necessary updates. **[8]**