UNIVERSITY OF LONDON

GOLDSMITHS COLLEGE

B.Sc. Examination 2013

COMPUTING AND INFORMATION SYSTEMS

IS53011A Language Design and Implementation (Resit)

Duration: 2 hours 15 minutes

Date and time:

There are three questions in this paper. You should answer them all. Each question is marked out of 100. The marks for each part of a question are indicated at the end of the part in [.] brackets.

THIS PAPER MUST NOT BE REMOVED FROM THE EXAMINATION ROOM

Question 1.

a) Give the six main phases of a programming language compiler. **[3]**

b) Draw the algorithmic structure of the front end of a compiler. **[5]**

c) Define the notion of a regular expression over a given alphabet. **[6]**

d) Show six initial strings that can be generated by the following regular expression:
$(a) \mid ((b)^* (c))$. **[6]**

e) Rewrite the following regular expression in a more compact format: $(a^* b^*)^*$. **[5]**

Question 2.

a) Design a nondeterministic finite state automaton (NFA) for the language:
$a\ (\ a\ |\ b\ )^*$ using Thompson's construction algorithm. **[6]**

b) Convert the NFA from part (a) above to a deterministic finite-state automaton (DFA) using the subset construction algorithm. **[9]**

c) Develop the transition graph and the transition table of the constructed deterministic finite-state automaton (DFA) from part (b). **[10]**

Question 3.

a) Explain the abbreviation LR (k) parsing used to denote a technique for bottom-up syntax analysis. **[5]**

b) Consider the following LR grammar and its parsing table:

(1) $S' \rightarrow S$
(2) $S \rightarrow FF$
(3, 4) $F \rightarrow xF \mid y$

	Action			Goto	
State	*x*	*y*	$	*S*	*F*
0	*s3*	*s4*		*1*	*2*
1			*acc*		
2	*s3*	*s4*			*5*
3	*s3*	*s4*			*6*
4	*r3*	*r3*	*r3*		
5			*r2*		
6	*r3*	*r3*	*r3*		

Demonstrate the operation of the LR parsing algorithm on the input: $x\ y\ x\ x\ y$ $, by demonstrating the contents of the stack, the input and the output. **[20]**

Question 4.

a) Given the following LL(*1*) grammar:

$$P \rightarrow \{\ S\ \}$$
$$S \rightarrow \mathbf{x} := E$$
$$E \rightarrow FE'$$
$$E' \rightarrow -FE' \mid +FE' \mid \in$$
$$F \rightarrow \mathbf{x} \mid \mathbf{y} \mid \mathbf{c}$$

Derive the functions *FIRST* and *FOLLOW* necessary for building the corresponding parsing table for implementing a top-down nonrecursive predictive parsing algorithm. **[5]**

b) Why do we need these functions *FIRST* and *FOLLOW* in top-down parsing? **[2]**

c) Suppose that the parsing table for the LL(*1*) grammar from part (a) is:

	x	**y**	**c**	+	-	:=	{	}	$
P							$P \rightarrow \{S\}$		
S	$S \rightarrow \mathbf{x}{:=}E$								
E	$E \rightarrow FE'$	$E \rightarrow FE'$	$E \rightarrow FE'$						
E'				$E' \rightarrow +FE'$	$E' \rightarrow -FE'$			$E' \rightarrow \in$	
F	$F \rightarrow \mathbf{x}$	$F \rightarrow \mathbf{y}$	$F \rightarrow \mathbf{c}$						

Illustrate the stack, the input and the output of the nonrecursive predictive parsing algorithm on the following input: { **x** := **y** – **c** + **x** }. **[18]**

Question 5.

Consider the following simple program which computes the greatest common divisor of two integers selected from a given array of numbers:

```
int gcd( int A[], int i, int j )
{
    int t;
    do
    {
        if ( A[ i ] < A[ j ] )
        { t = A[ i ]; A[ i ] = A[ j ]; A[ j ] = t; }
        A[ i ] = A[ I ] - A[ j ];
    }
    while ( A[ I ] > 0 );
    return A[ j ];
}
void main()
{
    int i = 3, j = 5;
    int A[] = { 1, 2, 3, 4, 5 };

    gcd( A, i, j );
}
```

Develop three-address intermediate code for this simple program fragment **[25]**