(CONFIDENTIAL)

BE 7th Semester Examination, 2011

Subject : Compiler Design **Paper/Code No CS 703**

Branch: CST

Full Marks: 70

Answer Q. no. 1 and any four from the remaining questions.

1. (a) What are the advantages of a translator module? [2]

(b) What are the reasons behind writing a multi-pass compiler rather than a single-pass one? [2]

(c) What do you mean by an ambiguous grammar? Explain with the following grammar and parse tree, how ambiguity takes place for the string 3-2+1, where {expr,'digit} is the set of non-terminal symbols, expr is the start symbol of the grammar and {+, - , 0, 1, 2,...,9} is the set of terminal symbols of the grammar. [1+3]

expr→ expr + expr | expr – expr | digit
digit → 0|1|2|3|4|5|6|7|8|9

(d) Define four types of grammar. [4]

(e) Inserting a code-optimizer within a compiler increases compilation time. How would you justify the presence of code-optimizer within a compiler? [2]

2. (a) Construct the LR(0) finite control transition diagram for the grammar with the set of terminal symbols V_T = {x, y, z, a, q, #}, set of non-terminal symbols V_N = {S', S, A, B}, # being a special terminal symbol used as end-marker , with the set of production rules as follows: [5]

S'→ S #
S →xAy |xBy | xAz
A →aS | q
B →q

(b) Compute FIRST and FOLLOW of all non-terminal symbols except S'. Construct the LR parsing table and from the table tell whether the grammar is SLR or not . [(2+3)+4]

3. (a) Consider the following code fragment. Generate the three-address code for it. [2]

```
while c>d do
        {x= x+ y;
         if a < b then a = a + b; else c = c + d ;
         }
```

(b) Write the syntax directed translation schemes for the grammar rules if else, while and boolean expressions. [3 + 3 + 6]

4. (a) State the disadvantages of a top-down parser using brute-force approach. [3]
(b) How can left-recursion be removed from a grammar? [2]
(c) Define FIRST(α) and FOLLOW (A), $\alpha \in V^*$ and $A \in V_N$, where V is the alphabet and V_N is the set of non-terminal symbols. [3]
(d) Define LL(1) grammar. [2]
(e) Discuss about recursive descent parser. How does left-factoring help design recursive descent parser? [3 + 1]

5.(a) State the conditions to be fulfilled for common sub-expression elimination and loop optimization. [3]
(b) Explain with example how DAG can be used for common sub-expression Elimination. [5]
(c) Define available expression data flow property. Which purpose is this data flow property useful for? State data flow equations for available expression data flow property. [2+1+3]

6.(a) Define synthesized attribute and inherited attribute. What is an L-attributed definition. [2+2+2]

Explain how L-attributed definition will be implemented during predictive parsing while evaluating the expression 9-5+2 using the following grammar rules when{ +, - ,(,), num} is the set of terminal symbols and {E, R, T } is the set of non-terminal symbols and E is the start symbol of the grammar. [8]

E→ T R
R→ + T R | -T R
R→ ϵ
T→ (E)
T→ **num**

7. (a)What are the criteria required to be satisfied for static storage allocation. [2]
(b) Define activation record. What are the different parts of activation record? Explain the role of static link and display in activation record during execution by using a suitable program segment. [2 + 1 + (2+2)]
(c) How will display be created when a function at level j (j<>i) will be entered from a function at level i ? [5]