B.TECH/IT/5TH SEM/INFO 3132/2016

COMPILER DESIGN
(INFO 3132)

Time Allotted : 3 hrs **Full Marks : 70**

Figures out of the right margin indicate full marks.

Candidates are required to answer Group A and any 5 (five) from Group B to E, taking at least one from each group.

Candidates are required to give answer in their own words as far as practicable.

Group – A
(Multiple Choice Type Questions)

1. Choose the correct alternative for the following: **10 × 1 = 10**

(i) A compiler for a high level language that runs on one machine and produce code for different machine is called
(a) optimizing compiler (b) one pass compiler
(c) cross compiler (d) multipass compiler.

(ii) $(a + b)^*$=?
(a) $(a^* + b^*)^*$ (b) $a^* + b^*$
(c) $(a^*b^*)^*$ (d) both (a) and (c).

(iii) A grammar that produces more than one parse tree for some sentence is called
(a) ambiguous (b) unambiguous
(c) regular (d) none of these.

(iv) _____ is the process where the stream of characters making up the source program is read from left to right and grouped into tokens.
(a) Lexical analysis (b) Syntax analysis
(c) Code generation (d) None of the above.

(v) Synthesized attribute can be easily simulated by a
(a) LL grammar (b) ambiguous grammar
(c) LR grammar (d) None of the above.

(vi) For predictive parsing the grammar A→AA I (A) I ε is not suitable because
(a) the grammar is right recursive
(b) the grammar is left recursive
(c) the grammar is ambiguous
(d) the grammar is an operator grammar.

(vii) In a compiler, the data structure responsible for the management of information about variables and their attributes is
(a) Semantic stack (b) Parser table
(c) Symbol table (d) Abstract syntax-tree

(viii) Shift reduce parsers are
(a) top down parser
(b) bottom up parser
(c) may be top down or bottom up parser
(d) none of the above.

(ix) Syntax directed translation scheme is desirable because
(a) it is based on the syntax
(b) its description is independent of any implementation
(c) it is easy to modify
(d) all of these.

(x) Which of the following is not an intermediate code form?
(a) postfix notation (b) syntax trees
(c) three address codes (d) quadruples.

Group – B

2. (a) With a block diagram briefly explain different parts of a language – processing system.

(b) Convert the following regular expression to NFA then NFA to DFA (if possible minimize the no. of states) : (a | b)* a b b (a | b)* (Write also the used functions and algorithms).

5 + (3 + 4) = 12

3. (a) Write the differences between compiler and interpreter.

(b) What is the role of Lexical Analyzer in compilation process?

(c) Construct the DFA's directly from the following regular expressions: (i) (a | b)* (ii) (a* | b*)* (iii) ((ε | a) b*)* . Hence show that all these expressions are equivalent.

3 + 1 + 8 = 12

Group – C

4. (a) For the following grammar, construct the parsing table.(You may left-factor and eliminate left-recursion from your grammar first) :
S – > S + S | S S | (S) | S * | a. Hence show the moves made by the predictive parser on input (a + a) * a.

(b) Give a syntax directed definition to differnetiate expressions such as x * (3 * x + x * x) involving the operators + and *, the variable x, and constants. Assume that no simplification occurs, so that, for example, 3 * x will be translated into 3 * 1 + 0 * x.

(5 + 3) + 4 = 12

5. (a) Show that the following grammar
S –> A a | b A c | d c | b d a
A –> d
is LALR(1) but not SLR(1).

(b) What is Handle Pruning? Explain with an example.

9 + (1 + 2) = 12

Group – D

6. (a) Translate the following statements into a syntax tree, quadruples, triples and indirect triples.
(i) x = a + -(b + c) + (b + c) (ii) a[i] = b * -c – b * d – b * c.

(b) What is Dangling Reference? Explain it with an example.

(4 + 4) + (1 + 3) = 12

7. (a) Consider the following C code to compute the Fibonacci numbers recursively:

```
int f(int n) {
   int t, s;
   if (n < 2) return 1;
   s = f(n – 1);
   t = f(n – 2);
   return s + t;
   }
```

Suppose that the activation record for f includes the following elements in order : (return value, argument n, local s, local t). Let us assume that the initial value for the function call is for 5, i.e., f(5).
(i) Show the complete activation tree.

(ii) What does the stack and its activation records look like the first time f(1) and fifth time f(5) are about to return?

(b) Explain with examples the difference between call – by – value and call–by–reference.

(c) Explain in brief two implicit deallocation strategies.

(3 + 3) + 3 + 3 = 12

Group – E

8. Consider the following matrix multiplication code: **(4 × 3) = 12**

```
for (i = 0; i< 10; i++)
    for (j = 0; j< 10; j++){
        c[i][j] = 0.0;
        for (k = 0; k< 10; k++)
            c[i][j] = c[i][j] + a[i][k] * b[k][j];}
```

(i) Translate the program into three address statements.
(ii) Construct the flow graph for your code from (i)
(iii) Identify the three address statements from the flow graph corresponding to "c[i][j] = c[i][j] + a[i][k] * b[k][j]" and draw a DAG for it.
(iv) Convert the set of three address statements of (iii) to the machine code.

9. Write short notes on any four: **(4 × 3) = 12**
(i) Peephole optimization
(ii) Loop optimization
(iii) Dead code elimination
(iv) Common subexpression elimination
(v) Global register allocation.