B.Tech Third Year End Semester Examination
Department: Computer Science and Engineering
Course Name: Compilers **Code: CS 346**
Full Marks-100 **Time: 3 hours**

Answer ALL the Questions

Make reasonable assumptions as and whenever necessary. You can attempt the questions in any sequence, but the answers to all the components of any particular question should appear together. Marks will be deducted if this is not followed properly.

1. (a). Consider the following segment of code:

```
if x + 5 > y and x < z then
 a = y * 10
else
 a = x
```

Assuming that numerical representation is being used for booleans, show three-address code to implement this segment (*Use the value 1 to denote true and the value 0 to denote false. Use statement numbers starting at 100. Assume that all variables are declared integers.*) **10**

(b). Can the graph coloring algorithm guarantees optimality for register allocation? Use graph coloring algorithm to generate the target code of the following set of instructions assuming only 2 registers are available. a := b + c, t1 := a * a , b := t1 + a , c := t1 * b, t2 := c + b , a := t2 + t2 **10**

2. (a). Consider the expression: E → E_1 or E_2| E_1 and E_2 | not E_1| (E_1) | id_1 relop id_2| true| false

Translate the expression a==b &&(c==d || e==f). Design the appropriate translation scheme and then show the values of *truelist* and *falselist* for each sub-expression (*Note: use the principle of backpatching*) **12**

(b). Generate three-address code for the following instructions assuming that all array elements are integers taking four bytes each. **8**
(i). X= a[i]+1
(ii).X=a/(b+c)-d*(e+f)

3. Consider the following set of basic blocks

B1={d1: i=m-1, d2: j:=n, d3: a:=u1}, **B2**={d4:i=i+1, d5:j=j-1}, **B3**={d6:a=u2}, **B4**={d7:i=u2 }

The CFG has the following set of edges: B1→B2; B2→B3, B4; B3→B4;B4→B2.

Compute the **GEN**, **KILL**, **IN** and **OUT** sets for the reaching definition analysis. **20**

4. Consider the CFG as follows: S → R, R → RR, R → R | R, R → R*, R → ε, R → a, R → (R)

[*Note that | is a terminal symbol in the grammar*].

LR(1) is a much stronger parsing algorithm than SLR(1). Would using an LR(1) parser instead of the SLR(1) parser resolve the ambiguities? Why or why not? Prove by generating all the canonical LR(1) item sets, and constructing the corresponding parsing table(s). **20**

5. Choose the correct answer **10*2=20**

(i). In a bottom-up evaluation of a syntax directed definition, inherited attributes can (a) always be evaluated; (b) be evaluated if the definition is L-attributed; (c) be evaluated only if the definition has synthesized attributes; (d) never be evaluated.
(ii). The process of assigning load addresses to the various parts of the program and adjusting the code and data in the program to reflect the assigned addresses is called (a). assembly; (b) parsing; (c). relocation, (d). symbol resolution.
(iii). The grammar $A \rightarrow AA \mid (A) \mid \varepsilon$ is not suitable for predictive-parsing because the grammar is (a). ambiguous; (b) left-recursive ; (c) right-recurisve; (d) an operator-grammar.
(iv). Dynamic linking can cause security concerns because (a) Security is dynamic; (b) the path for searching dynamic libraries is not known till run time; (c) linking is insecure; (d) cryptographic procedures are not available for dynamic linking.
(v). Assume that the SLR parser for a grammar G has n1 states and the LALR parser for G has n2 states. The relationship between n1 and n2 is (a) n1 is necessarily less than n2; (b) n1 is necessarily equal to n2; (c) n1 is necessarily greater than n2; (d) none of the above.
(vi). An LALR(1) parser for a grammar can have shift-reduce (S-R) conflicts if and only if (a) the SLR(1) parser for G has S-R conflicts; (b) the LR(1) parser for G has S-R conflicts; (c) the LR(0) parser for G has S-R conflicts; (d) the LALR(1) parser for G has reduce-reduce conflicts.
(vii). Which of the following suffices to convert an arbitrary CFG to an LL(1) grammar? (a) removing left recursion alone; (b) factoring the grammar alone,; (c) removing left recursion and factoring the grammar ; (d) none of this.
(viii). Consider the grammar rule $E \rightarrow E1 - E2$ for arithmetic expressions. The code generated is targeted to a CPU having a single user register. The subtraction operation requires the first operand to be in the register. If E1 and E2 do not have any common sub expression, in order to get the shortest possible code (a) E1 should be evaluated first; (b) E2 should be evaluated first; (c) evaluation of E1 and E2 should necessarily be interleaved; (d) order of evaluation of E1 and E2 is of no consequence.
(ix). DAG representation of basic block allows (a) automatic detection of local common sub expressions; (b) automatic detection of induction variables; (c) automatic detection of loop variants; (d). none of the above.
(x). What data structure in a complier is used for managing information about variables and their attributes? (a) abstract syntax tree; (b) symbol table; (c) semantic stack; (d) parse table