Compiler Design (CS - 701)
Assignment

1. Consider the following grammar G for Boolean expressions:

$$B \rightarrow B \; or \; T \mid T$$
$$T \rightarrow T \; and \; F \mid F$$
$$F \rightarrow not \; B \mid (B) \mid true \mid false$$

where B is the start symbol, set of non-terminals are $\{B, T, F\}$ and set of terminals are {*and, or, not, (,), true, false* }.

   (a) Compute $FIRST$ and $FOLLOW$ for each nonterminal in G.

   (b) Construct a predictive parsing table for G.

   (c) Show how your predictive parser processes the input string:

$$true \; and \; not \; false \; or \; true$$

   Draw the parse tree traced out by your parser.

2. (a) Describe the relationship between a production and an item in an LR(0) grammar.

   (b) Consider the following grammar and construct the set of LR(0) items. Then fill out an SLR parse table for this grammar and indicate whether the grammar is ambiguous.

$$S \rightarrow Aa \mid bAc \mid Bc \mid bBa$$
$$A \rightarrow d$$
$$B \rightarrow d$$

   Here S is the start symbol.

3. (a) Suggest a suitable approach for computing hash function for managing Symbol Table.

4. The following context-free grammar describes part of the syntax of a simple programming language. Nonterminals are given in capitals and terminals in lower case. VAR represents a variable name and CONST represents a constant. The productions for ASSN-STMT are not given, as they do not affect the question.

   $PROGRAM \rightarrow Procedure \; STMT\text{-}LIST$

   $STMT\text{-}LIST \rightarrow STMT \; STMT\text{-}LIST \mid STMT$

   $STMT \rightarrow do \; VAR = CONST \; to \; CONST \; \{STMT\text{-}LIST\} \mid ASSN\text{-}STMT$

   (a) Show the parse tree for the following:

```
Procedure
do i = 1 to 100 {
ASSN–STMT
ASSN–STMT
}
ASSN–STMT
```

   (b) Create attribute(s) and add semantic functions to the above grammar that compute the number of executed statements for a program conforming to this grammar. Write them beside the productions above.

   (c) Using the tree from part a, compute the value of your attributes.

   (d) Replacing $ASSN\text{-}STMT$ with the terminal a, in the above grammar, create lex and yacc files that will compute the number of occurrences of this terminal a. Use yacc's semantic facilities ($$ etc.).

5. (a) What do you mean by machine dependent and machine independent optimization?

6. (a) Consider the following program.

```
L0: e = 0
    b = 1
    d = 2
L1: a = b + 2
    c = d + 5
    e = e + c
    f = a * a
    if f < c goto L3
L2: e = e+f
    goto L4
L3: e=e+2
L4: d = d+4
    b=b-4
    if b != d goto L1
```

This program uses six temporaries, $a - f$. Assume that the only variable that is live on exit from this program is e. Draw the register interference graph.