Ref. No. Ex / CSE / T / 321 / 85 / 2016

BCSE 3RD YEAR EXAMINATION 2016

(2nd Semester)

Compiler Design

Time: Three Hours

Full Marks 100

Answer question no. 1 and any four from the rest

1. Answer any six questions. 20
   a) Define NFA and DFA.
   b) Clearly explain the difference between a context-free and context-sensitive grammar.
   c) With examples explain the difference between "*useless code elimination*" and "*unreachable code elimination*".
   d) Draw an abstract syntax tree for the expression y = 2 * x + sin(x) - 2
   e) What is recursive-descent parsing?
   f) Give an example of a control flow graph and three-address code for a "if-then-else" statement.
   g) What do you mean by handle pruning? Give example.
   h) What do you mean by peep-hole optimisation and local optimisation.

2. (a) Discuss the different stages of a compiler and for each stage clearly indicate its input and output.
   (b) What is the role of a parser?
   (c) What is a symbol table? A linked list based symbol table is to be constructed for the following program segment. Show the implementation of the symbol table and clearly explain the contents of the table at the beginning and the end of each block.

```
int a, b;
int sm (int size)
{       .....
char a, temp;
.....
        {       double b;
                .....
        }
        {       char b, c;
                .....
        }
}
```

5+3+(2+10)=20

3. (a) Define regular grammar. How can a language based on regular grammar be recognized?
   (b) Write regular expressions for the following statements and convert each of them to an NFA:
   i) Strings that start with an 'a' followed by zero or more 'b's or 'c's.
   ii) Strings that start with a '$' which is followed by exactly two decimal digits from 1 to 4.
   (c) Convert any one of the NFAs that you obtained from the question no. 3.(c) to a DFA.

2+8+10=20

4. (a) Write an algorithm for removing **immediate** left recursion.
   (b) Construct the LL(1) parsing table for the following grammar.

S --> a S b S | b S a S | ε
(Terminals = {a, b}, Non-terminal = {S}, Start Symbol = S)
(c) Write two strings (at least six characters long) that can be generated by the grammar. Give the leftmost derivation for each string and draw their parse trees.

6+6+(2+6)=20

5. (a) Distinguish between backtracking parsing technique and predictive parsing technique.
(b) Consider the following grammar:
S --> *AA*
S --> *bc*
A --> *baA*
A --> *c*
Construct the DFA of LR(0) items for this grammar. (Terminals = {a,b,c}, Non-terminals = {S,A}, Start Symbol = S)
(c) Construct the SLR parsing table for the above grammar.
(d) Show the actions of the parser for the input string: (bacbac).

3+6+7+4=20

6. (a) Consider the following augmented grammar:
S' --> S
S --> C C
C --> c C | d
Construct the DFA of LR(1) items (for canonical parsing) for this grammar.
(b) Construct a canonical parsing table for the above grammar. (Terminals = {c,d}, Non-terminals = {S', S, C}, Start Symbol = S').
(c) Consider the grammar:
decl --> *type* *var-list*
type --> int | float
var-list --> id , *var-list* | id
(i) What are the terminals and non-terminals of this grammar? (ii) Left factor this grammar, (iii) Construct the First and Follow sets for the non-terminals of the resulting grammar.

7+7+6=20

7. (a) What do you mean by syntax-directed transition? Define synthesized and inherited attributes.
(b) There is a production A --> BCD. Each of the non-terminals A,B,C, and D have two attributes: *s* is a synthesized attribute, and *i* is an inherited attribute. For each of the sets of rules below, mention whether the rules are consistent with S-attributed definition, or L-attributed definition or none of these two. **Justify your answer in each case.**
i) *A.s* = *B.i* + *C.s*
ii) *A.s* = *B.i* + *C.s* and *D.i* = *A.i* + *B.s*
iii) *A.s* = *B.s* + *D.s*
iv) *A.s* = *D.i*, *B.i* = *A.s* + *C.s*, *C.i* = *B.s* and *D.i* = *B.i* + *C.i*
(c) Write the three address code and static single assignment form of the following program fragment:

```
z = x * 2 + y * 2;
y = sum (y,z);
z = x * 4 + (a + b)/4;
x = x * x + y * y;
```

4+8+8=20