Ref. No. Ex/CSE/T/321/85/14

BCSE 3RD YEAR EXAMINATION 2014

(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) What are the different stages of a compiler?
b) Discuss Chomsky's hierarchy of grammar.
c) Define the following terms: token, pattern and lexeme.
d) Define NFA and DFA.
e) Give an example of each of the followings: Directed Acyclic Graph, Three-address code, Static Single Assignment.
f) Explain the terms: S-attributed definitions, L-attributed definitions.
g) Discuss the different scopes of code optimization.

2. (a) Write regular expressions to define
   (i) fixed decimal literals with no superfluous leading or trailing zeros.
   (ii) all strings of lowercase that begin and end in '*a*'.
(b) Convert the regular expression (a | (bc) * d) into an NFA.
(c) Convert the NFA into a DFA using the subset construction.
(d) Define context free grammar and explain with suitable example why it is different from context sensitive grammar.

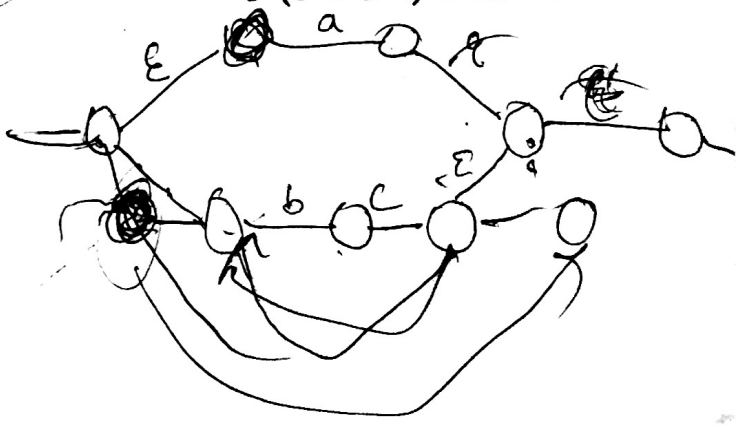
4+6+6+4=20

3. (a) What is parsing? Briefly explain the techniques (i) recursive descent parsing, and (ii) predictive parsing along with their differences.
(b) Consider the grammar:

decl --> *type* *var-list*
type --> **int** | **float**
var-list --> **id** , *var-list* | **id**

(Note – non-terminals are written with *italicized* font and terminals are written with **bold** font)
(i) Left factor this grammar, (ii) Construct the First and Follow sets for the non-terminals of the resulting grammar, (iii) Show that the resulting grammar is LL(1), (iv) Construct the LL(1) parsing table for the resulting grammar.
(c) Draw a parse tree for the string : **int x, y, z**

5+(2+4+3+4)+2=20



4. Consider the following grammar:

lexp --> *atom* | *list*
atom --> **number** | **identifier**
list --> **(** *lexp-seq* **)**
lexp-seq --> *lexp-seq lexp* | *lexp*

(Note – non-terminals are written with *italicized* font and terminals are written with **bold** font)

(a) Give leftmost derivation and rightmost derivation of the input string **(a (b (2)))**.
(b) Remove the left recursion.
(c) Construct the First and Follow sets for the non-terminals of the resulting grammar,
(d) Construct the LL(1) parsing table for the resulting grammar.
(e) Show the actions of the parser for the input string: **(a (b (2)))**.

3+3+4+5+5=20

5. Consider the grammar:

S --> **id** | V := E
V --> **id**
E --> V | **n**

(a) Construct LR(0) item set for the above grammar.
(b) Construct the SLR parsing table for the above grammar.
(c) What kind of conflict do you find in the SLR parsing table? Explain why.
(d) Construct LR(1) item set and LALR parsing table. Do you still find a conflict? Justify your observation.
(e) Show the actions of the LALR parser for the input string: **a := 2**

4+4+2+7+3=20

6.

(a) What do you mean by Syntax-directed Translation? How is a translation scheme written for an inherited attribute?
(b) Consider the following grammar where numbers may be octal or decimal. A one-character suffix **o** (octal) or **d** (decimal) is used for this purpose. Using the two attributes *base* and *val* for num and digit, write the semantic rules for the following grammar:

based-num --> *num basechar*
basechar --> **o** | **d**
num --> *num digit* | *digit*
digit --> **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

(c) Consider the following grammar. Construct a SLR parsing table for the above grammar.

tuple --> **(** *list* **)**
list --> *list* **a**
list --> **a**

(d) Explain the terms: (i) *handle pruning*, (ii) *closure of an item set.*

4+4+8+4=20

7.

(a) Translate the arithmetic expression 2*a*b*c+a*b+c into:
(i) Abstract syntax tree, (ii) Quadruple, and (iii) Triple.

(b) Write the three-address code of the following program fragment:

```
if (x > y)
{
        f(x, 2);
}
else
        y = x + 4 * y + z;
```

(c) With an example of each of the following optimization techniques, explain their advantages.
(i) Procedure inlining, (ii) Loop fission, (iii) Dead code elimination, (iv) Loop unrolling.

(d) Discuss how *Control Flow Statements* in a program are translated into three-address codes.

4+4+8+4=20