Ref. No. Ex/CSE/T/321/85/2015

BCSE 3RD YEAR EXAMINATION 2015

(2nd Semester)

Compiler Design

Time: Three Hours | Full Marks 100

Answer question no. 1 and any four from the rest

1. Answer any five questions. 20

a) Discuss the role of a lexical analyser and a parser.

b) Define and explain context-sensitive grammar.

c) What is a transition table? Give an appropriate example of a transition table.

d) Explain how you implement three-address code.

e) What is recursive-descent parsing? When do you need backtracking in case of parsing?

f) Discuss the advantages of Loop Interchange transformation. Is it always legal?

g) Briefly discuss the structure of a lex file.

2. (a) Write regular expressions to define

i) Integers divisible by five (i.e., 5, 10, 15, 45 and so on).

ii) Strings containing the characters {$, c, ., 0, ... 9} that are legal descriptions of monetary quantities. For example, $33, $5.32, 45c etc.

iii) Strings containing characters {a ... z} that are 4 characters long.

iv) Strings over alphabets {a ... z} containing at least one occurrence of the letter 'n'.

(b) Convert the regular expression (a|b)*ab into an NFA.

(c) Convert the regular expression (a|b)*a(a|b) directly to a DFA.

6+4+10=20

3. (a) What is the problem with a left-recursive grammar? Explain with an example. How do you convert a left-recursive grammar into a non left-recursive grammar?

(b) Consider the grammar:

$S \rightarrow ABC$

$A \rightarrow aA \mid \varepsilon$

$B \rightarrow bB \mid \varepsilon$

$C \rightarrow c$

(Terminals = {a, b, c}, Non-terminals = {S, A, B}, Start Symbol = S)

Construct the LL(1) parsing table for the grammar.

(c) Write three strings (at least four characters long) that can be generated by the grammar. Give the leftmost derivation for each string and draw their parse trees. 5+8+7=20

4. Consider the following grammar:

S --> AB
S --> BA
A --> aaB
A --> a
B --> bbA
B --> a

(Terminals = {a, b}, Non-terminals = {S, A, B}, Start Symbol = S)

(a) Construct LR(0) item set for the above grammar.

(b) Construct the SLR parsing table for the above grammar. Is the grammar SLR?

(c) Show the actions of the parser for the input string: **(aabba).** 6+7+7=20

5. Consider the following grammar for octal and decimal numbers. The base of a number is indicated by a one-character suffix 'o' (for octal) or 'd' (decimal).

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

(a) Write the semantic rules for the above grammar using two attributes 'val' and 'base' for num and digit. The rules must give the final value of *based-num*.

(b) Draw an annotated parse tree and a dependency graph for the above attribute grammar.

(c) What type of attributes are 'val' and 'base' (Synthesized or Inherited)? Can it be called an L-attributed definition?

(d) Explain the terms: S-attributed definitions, L-attributed definitions.

6+6+4+4=20

6. Consider the following grammar

S --> L=R | R

L --> *R | id

R --> L

(a) Construct an SLR parsing table for the above grammar. What kind of conflict you find in the table? Why?

(b) Construct a canonical parsing table for the same grammar.

10+10=20

7. (a) Discuss the following code optimization techniques. Give an example of each.

(i) Constant Folding, (ii) Loop Invariant Code Motion, (iii) Loop Unrolling, (iv) Useless Code Elimination and (v) Unreachable Code Elimination.

(b) What is a three-address code? What is a static single assignment form? Discuss the benefits of each.

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

```
while (x <= y)
        x = x * 2;
y = x + 4 * y + z;
```

(d) Write the three address code and static single assignment form of the following program fragment:

```
z = x * 2 + y * 2;
y = sin (z);
z = x * 4 + (a + b)/4;
x = cos (z);
x = x * x + y * y;
```

10+3+2+5=20