B. INFO. TECH 3Rd YEAR 1ST SEMESTER EXAMINATION, 2018

(1st Semester)

PRINCIPLES OF COMPILER DESIGN

Time: Three hours | Full Marks : 100

*Answer Question No. 1 and Attempt any **FOUR** questions from the rest*

1) Answer the following questions.

(a) Compute the Basic Blocks for the following code snippet

```
1)  i = 1
2)  j = 1
3)  t1 = 10 * i
4)  t2 = t1 + j
5)  t3 = 8 * t2
6)  t4 = t3 - 88
7)  a[t4] = 0.0
8)  j = j + 1
9)  if j <= 10 goto (3)
10) i = i + 1
11) if i <= 10 goto (2)
12) i = 1
13) t5 = i - 1
14) t6 = 88 * t5
15) a[t6] = 1.0
16) i = i + 1
17) if i <= 10 goto (13)
```

10

(b) Remove left recursion from the following grammar

$A \rightarrow B\,\mathbf{x}\,\mathbf{y} \mid \mathbf{x}$

$B \rightarrow C\,D$

$C \rightarrow A \mid \mathbf{c}$

$D \rightarrow \mathbf{d}$

4

(c) Is an LALR grammar always LR(1) grammar? Explain.

2

(d) What is an activation record? What are the contents of an activation record?

2+2

2) Construct a lexical analyzer (by constructing the RE, NFA, DFA, and the program) that recognizes tokens belonging to the following.

- Identifiers
  - begin with a letter, and continue with any number of letters
  - No identifier is longer than 4 characters
- Keyword (reserved): if
- Numbers:
  - any sequence of decimal digits, no sign
  - assume no number is longer than 4 digits

Other assumptions:

- The lexical analyzer is case insensitive
- Alphabet set is all English letters (upper and lower), digits, and blank
- No whitespace needed to separate tokens except when this changes the tokens (as cat dog vs catdog)

5+5+5+5

3) Answer the following questions.

a) Generate Three address code for the following pseudo Code

```
for i from 1 to 10 do
        for j from 1 to 10 do
                a[i, j] = 0.0;
for i from 1 to 10 do
        a[i, i] = 1.0;
```

b) Consider a grammar which can derive a code snippet as given in part (a).

S → for *id* from *num* to *num* do S ; | id = *num*;

In this grammar, only S is a variable or a non-terminal. All underlined symbol are terminal symbols. The italicized symbols are token as output from the lexical analyzer and hence may be considered terminal symbols for the grammar.

Write a translation scheme to generate three address codes for the above grammar.

10+10

4) Answer the following questions.

(a) Show the symbol table structure having scope information for the following code snippet.

```
f( ) {
    int m; float x; float y;
   {
      int i; int j;
   }
   {
       int x;
   }
}
g( ) {
    int m; bool t;
}
```

(b) Define a grammar to derive the code as shown in part (a).

(c) Define a syntax directed translation scheme for the grammar defined in part (b). You may use existing functions such as the following.

mktable(previous) returns a pointer to a new table that is linked to a previous table in the outer scope

enter(table, name, type, offset) creates a new entry in table

addwidth(table, width) accumulates the total width of all entries in table

enterproc(table, name, newtable) creates a new entry in table for procedure with local scope *newtable*

lookup(table, name) returns a pointer to the entry in the table for name by following linked tables

7+5+8

5) Consider the following grammar

$E \rightarrow T\ X$

$X \rightarrow +\ E \mid \varepsilon$

$T \rightarrow (\ E\ ) \mid \text{int}\ Y$

$Y \rightarrow *\ T \mid \varepsilon$

E, T, X, Y are variables or non-terminals.

b) Construct the First and the Follow sets.

c) Construct a LL(1) parse table for this grammar.

d) Parse the expression using the table. Clearly Show that steps of parsing.
int * int + int

5+8+7

6) Consider the following grammar

$A \rightarrow (A) \mid a$

$\sum = \{a, (,)\}$

Construct the SLR Parsing Table after augmenting it. There are 6 sets of items.

20

7) Answer the following questions..

a) Give an example of an ambiguous grammar and show that it is an ambiguous grammar.

b) What is meant by conflicts in LR parsing table (Explain using two types of conflicts)? If the LR(0) parsing table of a grammar has a conflict, how can you try to resolve that conflict (State just one possible way)?

c) Consider the following code segment

```
for (i = 0, i<n; i++) {
  for (j=0; j<n; j++)   {
    if (i%2)   {
      x += (4*j + 5*i);
      y += (7 + 4*j);
    }
  }
}
```

With respect to the above code snippet, answer the following questions:

i) What is common sub-expression that can be eliminated?

ii) What is the loop invariant computation?

iii) What is the scope of strength reduction?

iv) What is the possibility of dead code elimination?

(3+3)+(5+1)+(2+3+2+1)