Total No. of Questions : 12] [Total No. of Pages : 6

P1318

[3864]-404

B.E. (Computer Engineering)

PRINCIPLES OF COMPILER DESIGN

(2003 Course)

Time : 3 Hours] *[Max. Marks : 100*

Instructions to the candidates :

1) *Answers to the two sections should be written in separate books.*
2) *Neat diagrams must be drawn wherever necessary.*
3) *Figures to the right indicate full marks.*
4) *Assume suitable data, if necessary.*

SECTION - I

Q1) Write Lex specifications and auxiliary procedures to perform following tasks by reading the input text file. **[16]**

a) Count and display number of letters, words, vowels.

b) Copy the contents to an output file by

- Writing each sentence from the input file in "title case" with all vowels converted to uppercase and omitting all words that begin with letters x, y or z.
- Inserting a newline character after reading each full stop ('.') character.
- Insert a line number at the beginning of each line.
- Replace each letter 'v' with the letter 'w' if followed by the letter 'a'.
- Replace each letter 's' with the letter 'p' if preceded by the letter 'u'.

OR

Q2) a) In 'C' language, an integer constant can be expressed in decimal, octal or hexa-decimal number systems. An integer constant could be long or short. Write regular expressions to identify integer constants expressed in any of these forms. Construct DFA from these regular expressions. **[10]**

P.T.O.

b) Why it is not possible to design a lexical analysis tool that can detect the strings having equal number of two characters, say a and b. **[6]**

Q3) Construct LALR (1) parsing table for the following grammar. **[18]**

$D \rightarrow L : T$

$L \rightarrow L, id \mid id$

$T \rightarrow integer$

$T \rightarrow real$

$T \rightarrow$ array of [num DOTDOT num] of T

Show the moves of the parser for the strings.

i) a, b, c : integer

ii) p, q : array of [10 .. 20] of array of [20 .. 10] of real.

OR

Q4) a) Construct an LL(1) parsing table for the following grammar. Also compute the synchronization entries (that can be used for error recovery) **[10]**

$S \rightarrow aBDh$

$B \rightarrow cC$

$C \rightarrow bC \mid \varepsilon$

$D \rightarrow EF$

$E \rightarrow g \mid \varepsilon$

$F \rightarrow f \mid \varepsilon$

b) Prove that no left recursive grammar can be LL(1). **[4]**

c) Let X be a nullable nonterminal that derives to at least two terminal strings. Show that in a LL(1) grammar, no production rule can have two consecutive occurrences of symbol X on the right side of the production. **[4]**

Q5) a) Consider the following CFG.

$S \rightarrow$ for $(E_1; E_2; E_3)\ S_1$

Use marker non terminal(s), write syntax directed translation to translate the for statement into three address code statements. Describe the data structure and auxiliary functions needed. **[10]**

b) For the following source language statement, show the generated three address code statements using the translation scheme you have designed in Q. 5 (a) above. **[6]**

```
for (i = 1; i < = 20; i++)
      x = y + z
```

OR

Q6) a) Consider the following CFG.

$$\begin{aligned} S &\rightarrow \text{switch } E \ \{\text{caselist}\} \\ \text{caselist} &\rightarrow \text{caselist case } V : S \\ \text{caselist} &\rightarrow \text{case } V : S \\ \text{caselist} &\rightarrow \text{default} : S \\ \text{caselist} &\rightarrow \text{caselist default} : S \end{aligned}$$

Use marker non terminal(s), write syntax directed translation to translate the for statement into three address code statements. Describe the data structure and auxiliary functions needed. **[10]**

b) For the following source language statement, show the generated three address code statements using the translation scheme you have designed in Q. 6 (a) above. **[6]**

```
switch (a + b)
{
      case 2: {x = y; break;}
      case 5: {switch x
               {
                     case 0: {a = b + 1; break;}
                     case 1: {a = b + 3; break;}
                     default: {a = 2;    break;}
               }
               break;
      case 9: {x = y – 1; break;}
      default: {a = 2;    break;}
}
```

SECTION - II

Q7) a) Explain with suitable example code fragments, various parameter passing methods. [8]

b) 'C' compiler uses static, stack and heap allocation methods for storage management. Explain these mechanisms using a suitable example 'C' program. [8]

OR

Q8) a) Explain the 'Display' mechanism used by the PASCAL compiler to provide access to non-local names with nested procedures. Compare this technique with other techniques. [8]

b) Explain storage management techniques that are used to support variable length arrays and to handle function calls with variable number of arguments. [8]

Q9) Construct a DAG for the following three-address code. [16]

i) $t_1 = a + b$

ii) $t_2 = c + d$

iii) $t_3 = e - t_2$

iv) $t_4 = t_1 - t_3$

Label the DAG using labeling algorithm and show the generated code. Assume there are only two registers R_0 and R_1 and the machine supports all necessary instructions.

OR

Q10) a) Consider the three address code and assumptions about the underlying machine such as registers and instructions as given in Q. 9 above. Assume all three address code statements given in Q. 9 above are in one single basic block. [8]

Using the straightforward code generation algorithm, for each of the three address code statements, show

i) the location L that describes the entity where required computation are to be performed.

ii) generated instruction(s).

iii) contents of the register and address descriptor.

iv) cost in terms of memory words.

What is the total size of the code in terms of number of memory words?

b) If three address code statements (ii) and (iii) as given in Q. 9 above are swapped, show the generated code as in question 10 (a) above. What is the total size of the code in terms of number of memory words? **[8]**

Q11) a) 'Loop Unrolling' involves replicating the body of the loop to reduce the required number of tests if number of iterations are constant. Apply the technique to the loop shown below (i) Once and (ii) twice and explain its purpose in code optimization. **[6]**

```
k = 200;
while (k > = 0)
{
        arr [k] = 0;
        k--;
}
```

b) 'Loop Jamming' is a technique that merges the bodies of two loops if the two loops have the same number of iterations and they use the same indices. Apply this technique to following code fragment and explain its role in optimization. **[6]**

```
for (I = 0; I < 10; I++)
            for (I = 0; I < 10; I++)
                        X[I, J] = 0;
            for (I = 0; I < 10; I++)
                        X[I, J] = 1;
```

c) Write a note on Peephole Optimization. **[6]**

OR

Q12) a) 'Loop Test Replacement' is a technique that replaces a loop termination test phrased in terms of one variable, by a test phrased in terms of another variable. Apply this technique to the following code fragment and explain its purpose in optimization.

```
            temp = 5;
            i = 1;
L10:        x = temp
            i = i + 1
            temp = temp + 5
            if (i < = 10)
                        goto L10;
```

[6]

b) 'Loop reversal' is a technique that reverses the order in which values are assigned to the index variable. Apply this technique to the following code fragment that finds the last occurrence of *x* in an integer array val[50]. Illustrate its significance in code optimization. [6]

```
for (i = 0; i < = 49; i++)
{
        k = 49 – i;
        if (val [k] == x)
                break;
}
```

c) Explain the following : [6]

i) Reaching definitions.

ii) Live variables.

iii) Available expressions.

□□□□