Ex/CSE/PC/B/T/323/2022

B.E. COMPUTER SCIENCE AND ENGINEERING
THIRD YEAR
SECOND SEMESTER EXAM 2022

Time : Three hours Subject: **Compiler Design**

Full Marks: 100

Group-1 (20 marks)

Answer the following questions:

1. What is the role of lexical analyzer in Compiler Design? With examples explain (i) patterns, (ii) lexemes, and (iii) tokens. 5

2. (a) Write a regular expression for validating quaternary constants (only digits 0, 1, 2 and 3 are allowed) in a C-like programming language based on the following rules: (a) it must have at least one digit, (b) no space, comma or any other special symbols are allowed, (c) it can be positive or negative, (d) it may be an integer constant containing digits between 0 and 3, but should not begin with a zero, (e) it may also be a floating point constant with a decimal point. Leading zeros are not allowed if there is a non-zero value before decimal point and no trailing zero is allowed after the decimal point.
(b) Construct an NFA for the regular expression using McNaughton-Yamada-Thompson algorithm. Show every steps for construction of the NFA.
(c) Construct a DFA from the above NFA. [3+6+6]

OR

2. (a) Based on Chomsky's hierarchy, discuss the difference between *Context free grammar* and *Context sensitive grammar*.
A binary constant in C starts with '0b' or '0B' followed by a sequence of 0s and 1s. One "_" can be written between numbers.
Examples: 0b00101100; 0b0001_1000; 0B0_1_0_1_0_1_0_1_0_1_0_1_0_1_0_1.
Write a regular expression for the binary constant. Construct a DFA directly from the regular expression. [3+2+10]

Group-2 (40 marks)

Answer any two questions from this group.

3. (a) While constructing a parsing table, why do you need to construct the FIRST and FOLLOW sets? Explain your answer with examples.
(b) Consider the grammar:

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

(i) Remove left recursion in the grammar.
(ii) Construct First and Follow sets for the non-terminals.
(iii)- Construct the LL(1) parsing table.
(iv) Given the input string (a (b (2)), show the parsing actions. [4+2+4+6+4]

4. Consider the 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) Write two sentences (with at least four symbols) that can be produced from this grammar.
(b) Construct LR(0) item set for the above grammar.
(c) Construct the SLR parsing table for the above grammar. Is the grammar SLR?
(d) What are viable prefixes? Show at least two viable prefixes in the grammar. [2+7+7+4=20]

5. (a) What is a handle in *shift-reduce* parsing? What is handle pruning?
(b) Consider the grammar:

$S \rightarrow S\,(\,S\,)\,|\,\varepsilon$

Write a string generated from this grammar with at least 5 symbols. Show the rightmost derivation of the string and indicate the handles to be used for shift-reduce parsing at each step.
(c) Generate the LR(1) item set for the above grammar.
(d) Construct the LALR parsing table.
(e) Show the actions of the parser for the input string which you have used in question 5.(b).

[3+3+5+5+4=20]

Group-3 (30 marks)

Answer any two questions from this group.

6. (a) Define S-attributed and L-attributed grammar. Why are they important in semantic analysis phase of a compiler?
(b) The following grammar generates binary numbers:

$S \rightarrow L$
$L \rightarrow L\,B \mid B$
$B \rightarrow 0 \mid 1$

Design an S-attributed definition SDD to compute S.val, the decimal-number value of an input string. Write a translation scheme for the above SDD.
(c) Using a bottom-up parser, show how you can evaluate the decimal value of the binary string 1011.

[5+5+5=15]

7. (a) What type of information is stored in a symbol table? Discuss how the scope rules are stored in a symbol table implemented as a multiple hash table?
(b) Consider the following code block:

```
int sum (int k) {
int a = k;
float area = 1.0;
for (j = 0; j < a; j++) {
   float k = 3.14;
   double area;
   area += k * j * j;
}
 print(area);  }
```

What is the output of the code block? With appropriate implementation show how the scope of the identifiers are maintained in the symbol table. What are the complexities of insert() and lookup() operations in your implementation?
(c) What is three-address code? Discuss different implementations of three-address code.

[4+6+5=15]

8. (a) Given a grammar

$S \rightarrow$ while (C) S for a 'while' loop,

write the syntax directed definition and a translation scheme to generate the three address code for the 'while' loop. Explain how the translation scheme will work.
(b) Consider the following code block:

```
int a[10][10], b[10][10], sum, i;
sum = 0;
for (i=0; i<10; i++)
    for (j=0; j<10; j++)
        sum += a[i][j]+b[i][j];
```

Write a three address code representation for the above code block.
(c) What is 'static single assignment'? How does it differ from three-address code? Is your TAC representation in question 8(b) a static single assignment representation? Justify. If the answer is 'no', then convert it to SSA representation.

[5+5+5=15]

Group-4 (10 marks)

Answer any one question

9. Consider the following code block:

```
a := b + c
d := -a
e := d + f
if (e>10) then
    f := 2 * e
else {
    b := d + e
    e := e − 1 }
b := f + c
```

(a) Draw a Control Flow Graph for the above code block.
(b) Represent each block as three address code and find the 'liveness' and 'next use' for each variable in each block (consider all variables, not just temporaries (if any)).
(c) Draw a Register Interference Graph for the above code block.

[2+5+3=10]

10. (a) Optimize the following code and discuss each optimization technique that you have applied stating their advantages:

```
int main() {
  int i, n, array[10], k=1;
  n=5;
  n=k;
  for( i =0; i < 4; i++) {
      array[n+i] = i*5; }
  k = print();
  if (n>=5) array[n] = 5;
  return 0;
}
int print(){
 printf("Hello World !\n");
 return 0;
}
```

(b) Briefly explain how 'Graph Coloring algorithm' can be used for register allocation.

5+5=10