Reg. No. :

Question Paper Code : 40916

B.E./B.Tech. DEGREE EXAMINATION, APRIL/MAY 2018

Sixth Semester

Computer Science and Engineering

CS 6660 – COMPILER DESIGN

(Common to Information Technology)

(Regulations 2013)

Time : Three Hours

Maximum : 100 Marks

Answer ALL questions

PART – A

(10×2=20 Marks)

1. State the two main parts of compilation and its function.
2. Describe the possible error recovery actions in lexical analyzer.
3. Apply the rules used to define a regular expression. Give example.
4. What do you mean by Handle Pruning ?
5. Summarize the merits and demerits of LALR parser.
6. Draw the activation tree for the following code.

```
int main ( )
{
   printf("Enter Your Name");
   scanf("%s", username);
   int show_data(username);
   printf("Press Any Key to Continue...");
   ...
   int show_data(char *user)
 {
   printf(" Your Name is %s", username);
   return 0;
 }
 }
```

7. How do you identify predictive parser and non-recursive predictive parser ?
8. Name different storage allocation strategies used in run time environment.
9. Mention various techniques used for loop optimization.
10. List out the primary structure preserving transformations on basic block.

PART – B (5×13=65 Marks)

11. a) i) Draw a diagram for the compilation of a machine language processing system. (5)

ii) Apply the analysis phases of compiler for the following assignment statement.
position := initial + rate * 60. (8)

(OR)

b) i) Show the transition diagram for relational operators and unsigned numbers. (8)

ii) Outline the construction tools can be used to implement various phases of a compiler. (5)

12. a) i) Considering the alphabet $\Sigma = \{0, 1\}$. Construct a Non-Deterministic-Finite Automata (NFA) using the Thompson construction that is able to recognize the sentences generated by the regular expression (1 * 01 * 0) * 1 *. (8)

ii) Illustrate how does LEX work ? (5)

(OR)

b) Consider the regular expression below which can be used as part of a specification of the definition of exponents in floating-point numbers. Assume that the alphabet consists of numeric digits ('0' through '9') and alphanumeric characters ('a' through 'z' and 'A' through 'Z') with the addition of a selected small set of punctuation and special characters. (say in this example only the characters '+' and '–' are relevant). Also, in this representation of regular expressions the character '.' denotes concatenation.

Exponent = (+ | – | ε). (E | e) . (digit) +

i) Derive an NFA capable of recognizing its language using the Thompson construction.

ii) Derive the DFA for the NFA found in a) above using the subset construction.

iii) Minimize the DFA found in (ii) above using the interactive refinement algorithm described in class. (13)

13. a) Consider the Context-Free Grammar (CFG) depicted below where "begin", "end" and "x" are all terminal symbols of the grammar and stat is considered the starting symbol for this grammar. Productions are numbered in parenthesis and you can abbreviate "begin" to "b" and "end" to "e" respectively.

Stat → Block
Block → begin Block end
Block → Body
Body → x

i) Compute the set of LR(1) items for this grammar and draw the corresponding DFA. Do not forget to augment the grammar with the initial production S → Start$ as the production (0).

ii) Construct the corresponding LR parsing table. (13)

(OR)

b) i) Consider the following CFG grammar over the non-terminals {X, Y, Z} and terminals {a, c, d} with the productions below and start symbol Z. (6)

$X \rightarrow a$
$X \rightarrow Y$
$Z \rightarrow d$
$Z \rightarrow X\ Y\ Z$
$Y \rightarrow c$
$Y \rightarrow \varepsilon$

Compute the FIRST and FOLLOW sets of every non-terminal and the set of non-terminals that are *nullable*.

ii) Consider the following CFG grammar, (7)

$S \rightarrow aABe$
$A \rightarrow Abc \mid b$
$B \rightarrow d$

where a, b, c, d, e are terminals, 'S' (start symbol), A and B are non-terminals.

a) Parse the sentence " **abbcde** " using right-most derivations.

b) Parse the sentence " **abbcde** " using left-most derivations.

c) Draw the parse tree.

14. a) i) Describe about the contents of activation record. (6)

ii) Create a parse trees for the following string : string **id + id – id**. Check whether the string is ambiguous or not. (7)

(OR)

b) i) Explain about various ways to pass a parameter in a function with example. (6)

ii) Construct a Syntax-Directed Translation scheme that translates arithmetic expressions from infix into postfix notation. Using semantic attributes for each of the grammar symbols and semantic rules, Evaluate the input : **3 * 4 + 5 * 2.** (7)

15. a) i) Determine the basic blocks of instructions, Control Flow Graph (CFG) and the CFG dominator tree for following the code. (7)

```
01            a = 1
02            b = 0
03    L0 :    a = a + 1
04            b = p + 1
05            if (a > b) goto L3
06    L1:     a = 3
07            if (b > a) goto L2
08            b = b + 1
09            goto L1
10    L2:     a = b
11            b = p + q
12            if (a > b) goto L0
13    L3:     t1 = p * q
14            t2 = t1 + b
15            return t2
```

ii) Construct a code sequence and DAG for the following syntax directed translation of the expression : **(a + b) – (e – (c + d))**. (6)

(OR)

b) i) Translate the following assignment statement into three address code.
D:= (a - b) * (a - c) + (a – c)
Apply code generation algorithm, generate a code sequence for the three address statement. **(7)**

ii) Summarize the issues arise during the design of code generator. **(6)**

PART – C **(1×15=15 Marks)**

16. a) Draw the symbol tables for each of the procedures in the following PASCAL code (including main) and show their nesting relationship by linking them via a pointer reference in the structure (or record) used to implement them in memory. Include the entries or fields for the local variables, arguments and any other information you find relevant for the purposes of code generation, such as its type and location at run-time.

```
procedure main
integer a, b, c;
procedure f1 (a, b);
integer a, b;
call f2(b, a);
end;
procedure f2(y,z);
integer y, z;
procedure f3(m,n);
integer m, n;
end;
procedure f4(m,n);
integer m, n;
end;
call f3(c,z);
call f4(c,z);
end;
...
call f1(a, b);
end;
```

(15)

(OR)

b) Consider the following grammar

$E \rightarrow E + E$
$E \rightarrow E * E$
$E \rightarrow (E)$
$E \rightarrow id$

i) Find the SLR parsing table for the given grammar.

ii) Parse the sentence : **(a +b) * c.** **(15)**