SLR-SC – 57

Seat No.	

Set P

B.Sc. (E.C.S.) – III (Semester – VI) (CGPA Pattern) Examination, 2018
COMPUTER SCIENCE (Paper – IV)
Compiler Construction

Day and Date : Tuesday, 3-4-2018 Max. Marks : 70
Time : 2.30 p.m. to 5.00 p.m.

Instructions *: 1)* ***All*** *questions are* ***compulsory****.*
2) Figures to the ***right*** *indicate* ***full*** *marks.*

1. Choose the correct alternative : **14**

1) In compilers generation of intermediate code based on an abstract machine model is useful because
   A) Syntax-directed translations can be written for intermediate code generation
   B) To generate code for real machines directly from high-level language programs is not possible
   C) Portability of the front end of the compiler is enhanced
   D) Implementation of lexical and syntax analysis is easier

2) We have the grammar $E \rightarrow E+n|E \times n|n$. The handles in the right-sentential form of the reduction for a sentence $n + n \times n$ are
   A) n, n + n and n + n × n
   B) n, E + n and E × n
   C) n, E + n and E + E × n
   D) n, E + n and E + n × n

3) The languages that need heap allocation in the runtime environment are
   A) Those that use global variables
   B) Those that use dynamic scoping
   C) Those that support recursion
   D) Those that allow dynamic data structure

P.T.O.

4) In some programming language, L denotes the set of letters and D denotes the set of digits. An identifier is permitted to be a letter followed by any number of letters or digits. The expression that defines an identifier is

A) $(L.D)^*$ B) $(L + D)^*$ C) L(L.D) D) $L (L + D)^*$

5) Which one of the following statement is true ?

A) Canonical LR parser is more powerful than LALR parser

B) SLR parser is more powerful than LALR

C) LALR parser is more powerful than canonical LR parser

D) SLR parser, canonical LR parser and LALR parser all have the same power

6) Consider the following C program :

```
int main (){/*line1*/
int i, n;/*line 2*/
for (i=0,i
```

While creating the object module, the compiler's response about Line No.3 is

A) Only syntax error B) No compilation error

C) Only lexical error D) Both lexical and syntax error

7) We have the translation scheme given below :

$S \rightarrow FR$

$R \rightarrow$ *E{print('*');R| $\in$

$E \rightarrow F+E\{print('+');|F$

$F \rightarrow (S)|$ id{print(id.value);}

In the above translation scheme id represents the token in integer form and id value represents the corresponding integer value. What will be printed by this translation scheme when an input is '2 * 3 + 4' ?

A) 2 3 * 4 + B) 2 3 4 +*

C) 2 * + 3 4 D) 2 * 3 + 4

8) For the expression grammar

$E \rightarrow E*F \mid F+E \mid F$

$F \rightarrow F- \mid$ id

The statement, which holds true, is

A) + and – have same precedence B) Precedence of * is higher +

C) Precedence of – is higher * D) Precedence of + is higher *

9) Which one of the following statements holds true for a bottom-up evaluation of syntax directed definition ?
   A) Inherited attributes can always be evaluated
   B) Inherited attributes can never be evaluated
   C) Inherited attributes can be evaluated only if the definition is L-attributed
   D) Inherited attributes can be evaluated only if the definition has synthesized attributes

10) For predictive parsing, the grammar $A \rightarrow AA \mid (A) \mid \in$ is not suitable because
   A) The grammar is right recursive
   B) The grammar is left recursive
   C) The grammar is ambiguous
   D) The grammar is an operator grammar

11) Assuming that the input is scanned in left to right order, while parsing an input string the top-down parser use
   A) Rightmost derivation
   B) Leftmost derivation
   C) Rightmost derivation that is traced out in reverse
   D) Leftmost derivation that is traced out in reverse

12) _____ is a top-down parser.
   A) Operator precedence parser
   B) An LALR (k) parser
   C) An LR (k) parser
   D) Recursive descent parser

13) Why is the code optimizations are carried out on the intermediate code ?
   A) Because for optimization information from the front end cannot be used
   B) Because program is more accurately analyzed on intermediate code than on machine code
   C) Because for optimization information from data flow analysis cannot be used
   D) Because they enhance the portability of the compiler to the other target processor

14) In a compiler, when is the keyboards of a language are recognized ?
   A) During the lexical analysis of a program
   B) During parsing of the program
   C) During the code generation
   D) During the data flow analysis

2. Answer the following (**any seven**) : **14**

   1) List the phases that constitute the front end of a compiler.
   2) What is meant by Handle and Handle Pruning ?
   3) Why lexical and syntax analyzers are separated out ?
   4) What is operator precedence parser ?
   5) What are the problems with top down parsing ?
   6) What is phrase level error recovery ?
   7) Mention the functions that are used in back-patching.
   8) What is a flow graph ?
   9) What is code motion ?

3. A) Answer the following (**any two**) : **10**

   1) Consider the following Context Free Grammar G = ({S, A, B}, S, {a, b}, P) where P is

      $S \rightarrow AaAb$

      $S \rightarrow Bb$

      $A \rightarrow \in$

      $B \rightarrow \in$

      a) Compute the FIRST sets for A, B and S.

      b) Compute the FOLLOW sets for A, B and S.

      c) Is the CFG G LL(1) ? Justify ?

   2) Define string. Give commonly used string related terms with example.
   3) What are the types of Parser ? Give some common programming errors with example which can occur at different levels.

   B) Consider the expression a + a * (b – c) + (b – c) * d. **4**

      a) Draw the Syntax Tree.

      b) Draw the DAG.

      c) Give the postfix notation for same.

      d) Give the code sequence for the same.

4. Answer the following (**any two**) : **14**

   1) Explain an Activation Record.

   2) Construct a table-based LL(1) predictive parser for the following grammar :

      G = {bexpr, {bexpr, bterm, bfactor}, {not, or, and, (,), true, false}, P} with P given below.

      bexpr $\rightarrow$ bexpr or bterm | bterm

      bterm $\rightarrow$ bterm and bfactor | bfactor

      bfactor $\rightarrow$ not bfactor | (bexpr) |true| false

      For this grammar, answer the following questions :

      a) Remove left recursion from G.

      b) Left factor the resulting grammar in (a).

      c) Compute the FIRST and FOLLOW sets for non-terminals.

      d) Construct the LL parsing table.

   3) Explain the primary structure preserving transformations and algebraic transformations on basic block with example.

5. Answer the following (**any two**) : **14**

   1) What is Shift-Reduce Parsing ? Consider the following grammar and input string. Parse the string using shift reduce parser. Show the content of the stack, input and action taken at each stage.

      $S \rightarrow aB|\ bA$

      $A \rightarrow bAA|aS|a$

      $B \rightarrow aBB|bS|b$

      Input string : aabbab

   2) Explain in detail Loops in Flow Graphs.

   3) What are the various methods of implementing three address statements ? Also give the types of three address statements.
