b) What do you mean by a Handle ? Give example. When a grammar is called ambiguous ? Is there any technique to remove ambiguity ? Explain with an example. What is the reduce-reduce conflict in LR parser ? What are the various data structures used for symbol table construction ? 10 + 5

8. Generate the three address code for the following code segment (show the semantic actions).

```
Main ( )
   {
   int a=1, z = 0;
   int b[10];
   While (a<=10)
          b[a] = 2*a;
          z=b[a] +a;
   }
```

9. Consider the following grammar :

$S \rightarrow aABb$

$A \rightarrow c \mid \varepsilon$

$A \rightarrow d \mid \varepsilon$

Prove the above grammar is LL (1).

Draw the parsing table.

Now check whether the string "ab" and "acdb" are the languages of the above grammar.

(Derive each step with the help of a stack.)

10. Consider the following grammar :

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow id$

Draw a SLR state transition diagram for the above grammar. Also draw the SLR parse table.

11. Write short notes on any *three* of the following : 3 × 5

a) LEX

b) YACC

c) Peephole optimization

d) Symbol Table

e) Cross compiler.

CS/B.Tech/(CSE-New)/SEM-7/CS-702/2013-14

2013

COMPILER DESIGN

Time Allotted : 3 Hours *Full Marks* : 70

The figures in the margin indicate full marks.

Candidates are required to give their answers in their own words as far as practicable.

GROUP – A

(Multiple Choice Type Questions)

1. Choose the alternative answers for the following : 10 × 1 = 10

i) Which of the following is the most powerful parser ?

a) SLR b) LALR

c) Canonical LR d) Operator precedence.

ii) Inherited attribute is a natural choice in

a) keeping track of variable declaration

b) checking for the correct use of *L* values and *R* values

c) both (a) and (b)

d) none of these.

iii) A top down parser generates

a) rightmost derivation

b) rightmost derivation in reverse

c) left most derivation

d) left most derivation in reverse.

iv) In operator precedence parsing, precedence relations are defined

a) for all pair of non-terminals

b) for all pair of terminals

c) to delimit the handle

d) only for a certain pair of terminals.

v) A parser with the valid prefix property is advantageous because it

a) detect errors as soon as possible

b) detect errors as and when they occur

c) limits the amount of erroneous output passed to the next phases

d) all of these phases.

vi) Which of the following is used for grouping of characters into tokens ?

a) Parser b) Code optimization

c) Code generator d) Lexical analyzer.

vii) Three-address code involves .

a) exactly 3 addresses b) at most 3 addresses

c) no unary operator d) none of these.

viii) .... or scanning is the process where the stream of characters making up the source program is read from left to right grouped into tokens.

a) Lexical analysis b) Diversion

c) Modeling d) None of these.

ix) The graph that shows basic blocks and their successor relationship is called

a) DAG b) Flow chart

c) Control graph d) Hamiltonian graph.

x) A compiler that runs on one machine and produces code for a different machine is called

a) Cross compilation b) One pass compilation

c) 2 pass compilation d) none of these.

GROUP – B

(Short Answer Type Questions)

Answer any *three* of the following. 3 × 5 = 15

2. Explain different stages of compiler with a suitable example. What is Token, Patter and Lexeme ?

3. Explain left factoring with suitable example.

4. Draw a Syntax tree and DAG from the following expression.

   If $x > 0$ then $x = 3 * (y + 1)$ else $y = y + 1$

5. Compare quadruples, triples and indirect triples.

   $x = (a + b)^* - c / d$

   Represent this expression in quadruples, triples and indirect triples form.

6. Convert the regular expression $(a + b)^* abb$ into e-NFA and then corresponding DFA.

GROUP – C

(Long Answer Type Questions)

Answer any *three* of the following. 3 × 15 = 45

7. a) What is Basic Block ? List out the basic blocks and draw the flow graph for the following code :

```
1.  location = – 1
2.  i = 0
3.  i < 100 goto 5
4.  goto 13
5.  t1 = 4 i
6.  t2 = A [ t1 ]
7.  if t2 = x goto 9
8.  goto 10
9.  location = i
10. t3 = i + 1
11. i = t3
12. goto 3
13. ..
```