LR parsers can handle _____ grammars.
Left-recursive Page no: 163
file-recursive
End-recursive
Start-recursive

_____ convert the reloadable machine code into absolute machine code by linking library and reloadable object files.

Assembler

Loader/link-editor

Compiler

Preprocessor

Consider the grammar A --> B C D

B --> h B | epsilon

C --> C g | g | C h | i

D --> A B | epsilon

Follow of B is _____ .

h

g, h, i, $

g, i

g

Consider the grammar A --> B C D

B --> h B | epsilon

C --> C g | g | C h | i

D --> A B | epsilon

Follow of C is _____ .

g, h, i, $ Page no : 47

g, h, $

h, i, $

h, g, $

An important component of semantic analysis is _____ .
code checking
type checking page no : 6
flush checking
None of the given

In PASCAL _____ represent the inequality test.
:=
=
<>
None of the given

Lexical Analyzer generator _____ is written in Java.

Flex

Jlex Page no : 26

Complex

None of given

_____avoid hardware stalls and interlocks.

Register allocation

Instruction scheduling Page no : 10

Instruction selection

None of given

Consider the following grammar,

A --> B C D

B --> h B | episilon

C --> C g | g | C h | i

D --> A B | episilon

First of A is _____ .
h, g, i
g
h
None of the given

Recursive _____ parsing is done for LL(1) grammar.
Decent Page no : 47
Ascent
Forward
Backward

One of the core tasks of compiler is to generate fast and compact executable code.

True
False

Left factoring of a grammar is done to save the parser from back tracking.
True Page no:61
False

Responsibility of _____ is to produce fast and compact code.
Instruction selection Page no: 9
Register allocation
Instruction scheduling
None of given

_____ algorithm is used in DFA minimization.
James's
Robert's
Hopcroft's Page no:25
None of given

Compilers are sometimes classified as.

Single pass

Multi pass

Load and go

All of the given

In multi pass compiler during the first pass it gathers information about _____ .
Select correct option:
Declaration
Bindings
Static information
None of the given **

Question # 9 of 10 (Start time: 06:40:30 PM) Total Marks: 1
Flex is an automated tool that is used to get the minimized DFA (scanner).
Select correct option:
True
False

In compilation process Hierarchical analysis is also called
Select correct option:
Parsing
Syntax analysis
Both Parsing and Syntax analysis
None of given

For each language to make LL(1) grammar, we take two steps, 1st is removing left recurrence and 2nd is applying fin sequence.

True
False

_____is evaluated to yield a value.

Command
Expression
Declaration
Declaration and Command

LR parsers can handle _____ grammars.
Left-recursive page no : 63
file-recursive
End-recursive
Start-recursive


Optimal registers allocation is an NP-hard problem.
True
False Page no : 10

Parser takes tokens from scanner and tries to generate _____ .

Binary Search tree
Parse tree
Syntax trace Page no : 6
None of the given

Front end of two pass compiler takes_____ as input.

Source code Page no: 5

Intermediate Representation (IR)

Machine Code

None of the Given

In DFA minimization we construct one _____ for each group of states from the initial DFA.
State Page no : 25
NFA
PDA
None of given

In Three-pass compiler _____ is used for code improvement or optimization.

Front End

Middle End **Page no : 10**

Back End

Both Front end and Back end

_____ of a two-pass compiler is consists of Instruction selection, Register allocation and Instruction scheduling.

Back end **Page no : 9**

Front end

Start

None of given

NFA is easy to implement as compared to DFA.

True

False **Page no : 19**

We can get an LL(1) grammar by _____ .

Removing left recurrence

Applying left factoring

Removing left recurrence and Applying left factoring

None of the given

Parser always gives a tree like structure as output

True **page no : 37**

False

Intermediate Representation (IR) stores the value of its operand in

Registers

Memory

Hard disk

Secondary storage

In Back End module of compiler, optimal register allocation uses_____ .

O(log n)

O(n log n)

N P-Complete **Page no : 10**

None of the given

CS 606 Quizez

Can a DFA simulate NFA?

Yes

No

Sometimes

Depend upon nfa

_____ phase which supports macro substitution and conditional compilation.

Semantic

Syntax

Preprocessing

None

Which of the statement is true about Regular Languages?

Regular Languages are the most popular for specifying tokens.

Regular Languages are based on simple and useful theory.

Regular Languages are easy to understand.

All of the given

Lexer and scanner are two different phases of compiler

True

False Page no :13

Lexical Analyzer generator _____ is written in Java.

Flex

Jlex Page no :26

Complex

None of the given

In a transition table cells of the table contain the _____ state.

Reject state

Next state Page no 18

Previous state

None of the given

The transition graph for an NFA that recognizes the language (a | b)*abb will have following set of states.

{0}

{0,1}

{0,1,2}

{0,1,2,3} not sure

Front end of two pass compiler takes_____ as input.

Source code

Intermediate representation

Machine code

None

Functions of Lexical analyzer are?

Removing white space

Removing constants, identifiers and keywords

Removing comments

All of the given

Question # 1 of 10 (Start time: 07:25:59 PM) Total Marks: 1

Front-end of a two pass compiler is consists of Scanner.

Select correct option:

True

False

Question # 2 of 10 (Start time: 07:26:40 PM) Total Marks: 1

LL(1) parsing is called non-predictive parsing.

Select correct option:

True

False

Question # 3 of 10 (Start time: 07:28:09 PM) Total Marks: 1

Recursive _____ parsing is done for LL(1) grammar.

Select correct option:

Backward

Forward

Ascent

Decent

Question # 4 of 10 (Start time: 07:29:35 PM) Total Marks: 1

In predictive parsing table the rows are _____ .

Select correct option:

Non-terminals

Terminals

Both non-terminal and terminals

None of the given

Question # 5 of 10 (Start time: 07:30:38 PM) Total Marks: 1

We can get an LL(1) grammar by _____ .

Select correct option:

Removing left recurrence

Applying left factoring

Removing left recurrence and Applying left factoring

None of the given

Question # 6 of 10 (Start time: 07:31:48 PM) Total Marks: 1

_____ of a two-pass compiler is consists of Instruction selection, Register allocation and Instruction scheduling.

Select correct option:

Backend

Frontend

Start

Question # 7 of 10 (Start time: 07:32:26 PM) Total Marks: 1

Consider the grammar

A --> B C D

B --> h B | epsilon

C --> C g | g | C h | i

D --> A B | epsilon First of C is _____ .

Select correct option:

g, I look down at for reference

g

h i

i

Quiz Start Time: 07:25 PM

Time Left 81

sec(s)

Question # 8 of 10 (Start time: 07:33:12 PM) Total Marks: 1

Alternative of the backtrack in parser is Look ahead symbol in _____ .

Select correct option:

input

output

input and output

none

Question # 9 of 10 (Start time: 07:34:09 PM) Total Marks: 1

AST summarizes the grammatical structure with the details of derivations.

Select correct option:

True

False

Quiz Start Time: 07:25 PM

Time Left 81

sec(s)

Question # 10 of 10 (Start time: 07:35:06 PM) Total Marks: 1

One of the core tasks of compiler is to generate fast and compact executable code.

Select correct option:

True

False

Left factoring is enough to make LL1 grammar

True

False

In LL1() parsing algorithm _____ contains a sequence of grammar symbols.

Stack

Link list

Array

None

Question No 1

What is the role of Automatic Code Generator with respect to compiler?

Answer:

It generates efficient tokens automatically. It is known as Lexer generator.

Question No2

How Linker play an important role with respects to compilers constructions?

Answer:

A linker or link editor is a computer program that takes one or more source files generated by a compiler and combines them into a single executable program.

Question No3

Consider the grammar for arithmetic expressions?

E→ id Q

Q→ ER| ε

R→ +Q | -Q| *Q| /Q

Show the follow set for all the non-terminal in the grammar

Follow{E}= ?

Solution:

Rules for making follow set:

1. First put $ (the end of input marker) in Follow(S) (S is the start symbol)
2. If there is a production A → aBb, (where a can be a whole string) **then** everything in FIRST(b) except for ε is placed in FOLLOW(B).
3. If there is a production A → aB, **then** everything in FOLLOW(A) is in FOLLOW(B)
4. If there is a production A → aBb, where FIRST(b) contains ε, **then** everything in FOLLOW(A) is in FOLLOW(B)

Here a and b is terminal while A,B are non terminals.

In our question

E→ id Q

Q→ ER| ε

Follow{Q}={+,-,*,/}

R→ +Q | -Q| *Q| /Q

Id, ε, + , -, * , / are terminals

E, Q, R are non terminals.

Follow{E}={+,-,*,/,$}

Question No 4

Consider the following grammar. Calculate the first set of non- terminal S, A and B

S⟶ AB

A⟶ a | ε

B⟶ b | ε

Solution:

S⟶ AB

Rules for making first set:

1) If S→ A….Z then first of S will be First of A….Z
2) If First of A….Z all contains ε then first of S will also contain ε
3) If any (single) First set from A….Z $_{doesn't}$ contain ε in their First set then First of S will also doesn't contain ε.
4) The first of A will be first non terminal on r.h.s
5) The first of B will also be the non terminal on r.h.s
6) Note: ε is also a terminal.

Now make first sets for S,A and B

First{A}={a, ε }

First{B}={b ,ε}

Question NO: 5

Consider the grammar

A → B C D

B → h B | ε

C → C g | g | C h | i

D → A B |ε

Answer:

Rules for making first set:

1) If S→ A….Z then first of S will be First of A….Z
2) If First of A….Z all contains ε then first of S will also contain ε
3) If any (single) First set from A….Z $_{doesn't}$ contain ε in their First set then First of S will also doesn't contain ε.

4) The first of A will be first non terminal on r.h.s
5) The first of B will also be the non terminal on r.h.s
6) Note: ε is also a terminal.

A → B C D

B → h B | ε

C → C g | g | C h | i

D → A B |ε

First {A}={h,g,i}

Note: As the production C didn't contain ε so we didn't add ε in First {A}.

First {B}={h, ε}

First{C}={g,i}

First{D}={ h,g,i, ε }

Note: **We added** ε there because D originally has ε as a terminal in it.

Q. In two pass assembler what is the objective of First Pass? 2 marks

Answer:

First pass contains Scanner + Parser in it. Scanner takes the streams of chcracters as input and converts them into words or tokens

<Token type, word>

While Parser check those tokens for syntactic analysis and semantic analysis if found errors then report them after that it develop an IR.

Q. What is Register allocation? 3 marks

Answer:

Registers in a CPU play an important role for providing high speed access to operands. Memory access is an order of magnitude slower than register access. The back end attempts to have each operand value in a register when it is used. However, the back end has to manage a limited set of resources when it comes to the register file. The number of registers is small and some registers are pre-allocated for specialize used, e.g., program counter, and thus are not available for use to the back end. Optimal register allocation is NP-Complete.

Q. Up frontier discovery useful for top down parser.

Answer:

A bottom-up parser builds the parse tree starting with its leaves and working toward its root. The upper edge of this partially constructed parse tree is called its upper frontier

Parser tree 3 marks

Answer:

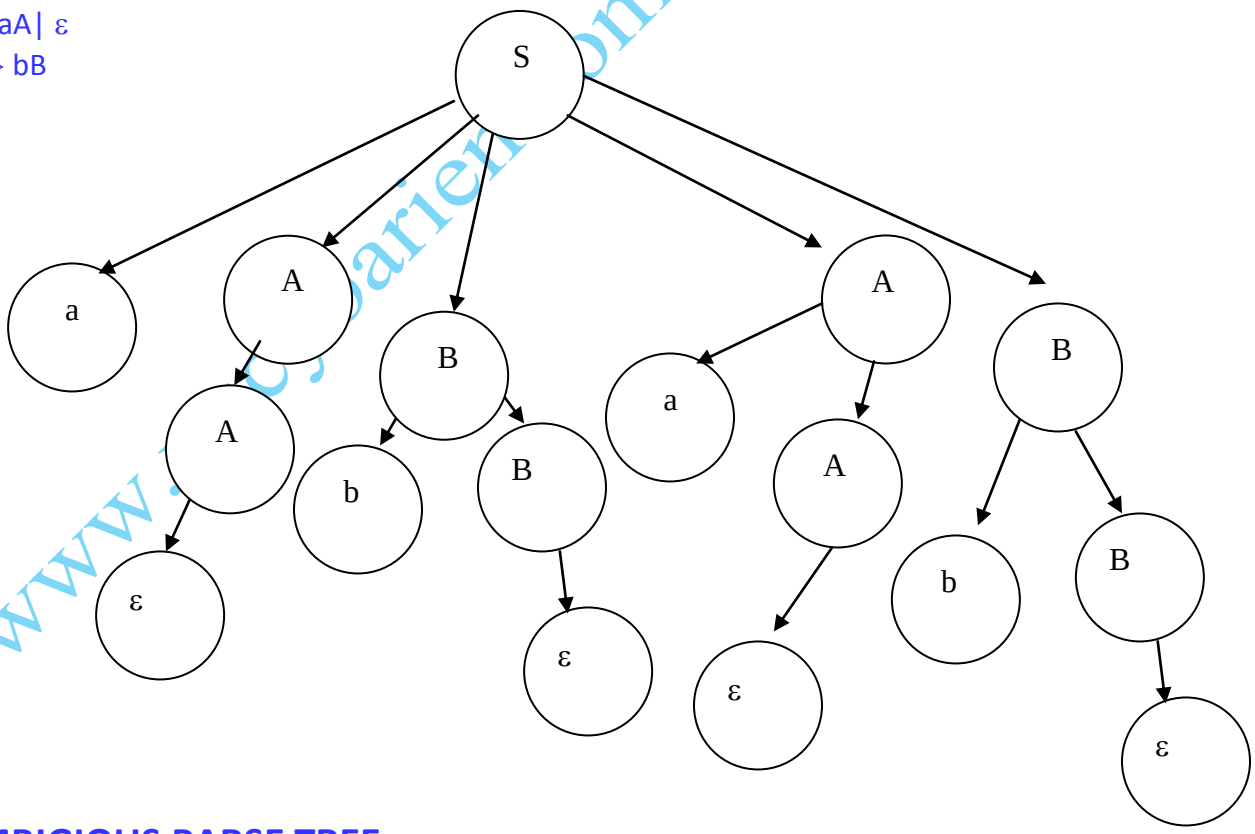
A parse can be represented by a tree: parse tree or syntax tree.

Q1)Considering string "ab", show that the following grammar is ambiguous using parse trees. The ε below is epsilon.

S → A B A B
A → a A | ε
B → b B | ε

Solution:

S→ABAB
A→aA| ε
B–> bB

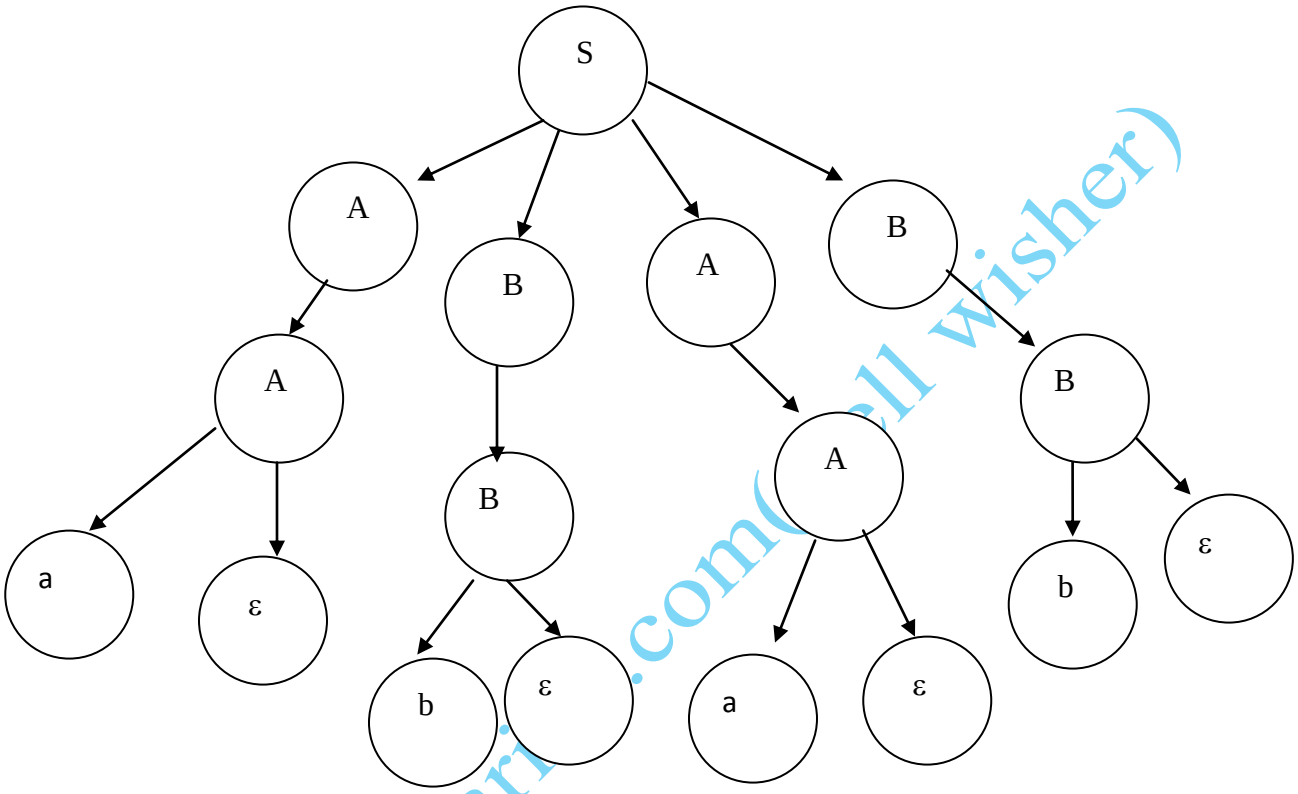


AMBIGIOUS PARSE TREE:

Can be written as

S→ABAB
A→ a ε
B--> b ε

Here it create ambiguity the string “ab” can be re written with internal A



Θ2) vConsider the following grammar;

Statement → if expression then statement else statement
Statement → if expression then statement

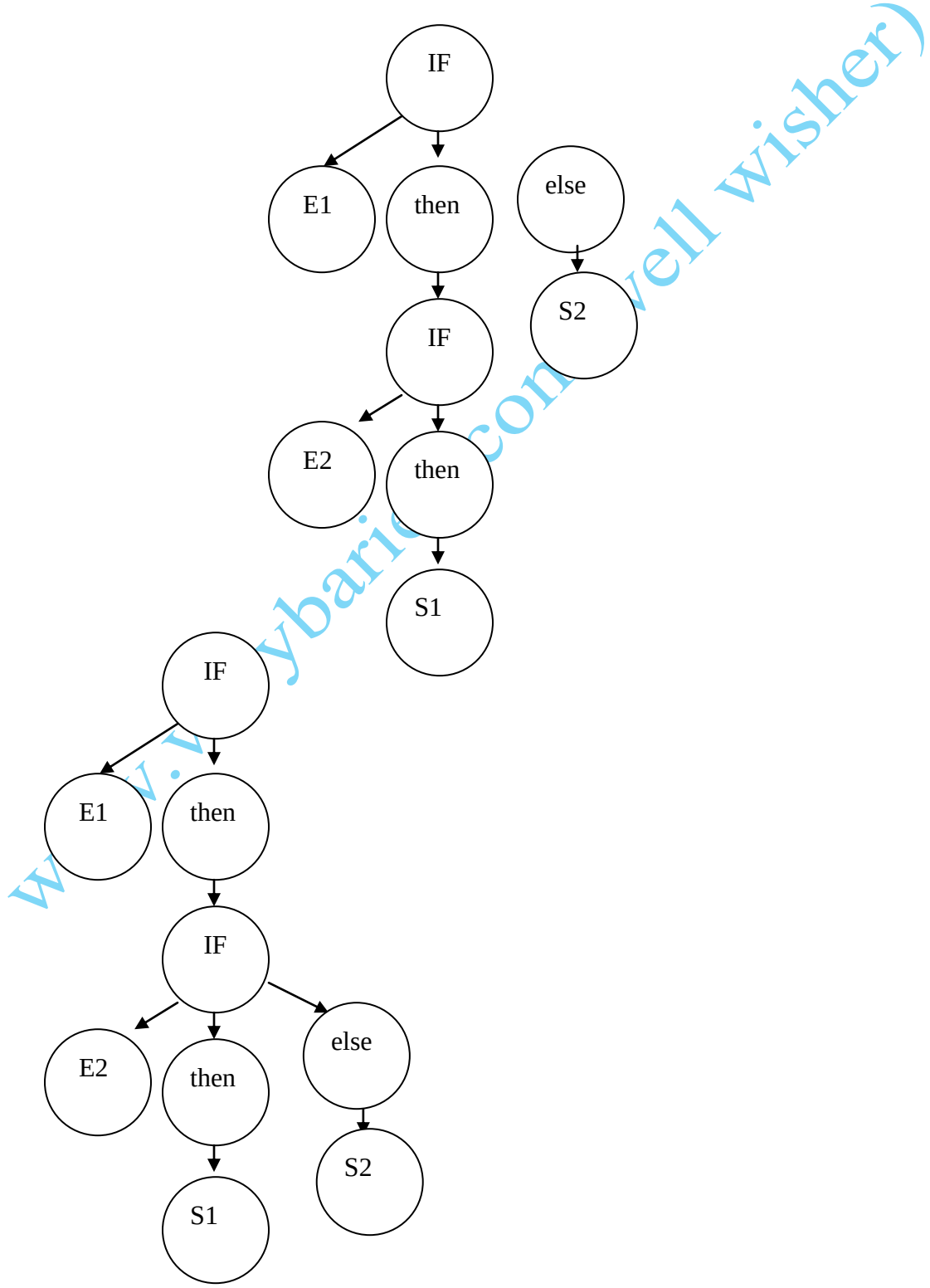
You are required to provide an alternate production(s) so that it may become free from the backtracking.
(5Marks)
Solution:

Statement → if expression then statement

Statement → if expression then statement else statement

Solution:

IF E1 then IF E2 then S1 Else S2



This is the ambiguous situation where

1)

IF E1
Then
  IF E2
  S1
Else
  S2

2)

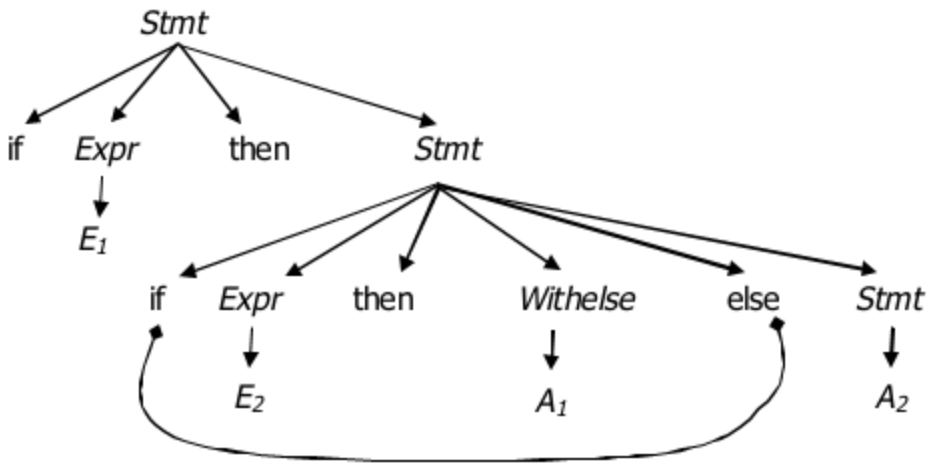
We can derivate this grammar in another way also

IF E1
Then
  IF E2
    S1
  Else
    S2

Here to correct it out we introduce another grammar for same situation by including one more NT.

Stmnt→ if E1 then stmnt
  | If E1 then with else else stmnt
  | Assignment
Stmnt→ if E2 then with else else stmnt
  | assignment

IF E1
Then
  IF E2
  A1
  Else
  A2



Q4) How recursive descent parser is implemented using any Object Oriented language?

(3Marks)

Answer:

We can use C++ to code the recursive descent parser in an object oriented manner. We associate a C++ class with each non-terminal symbol. An instantiated object of a non-terminal class contains pointer to the parse tree.

Q6) Write a regular expression for the language of all words that starts and ends with different letters.

(2 Marks)

Answer:

a(a+b)*b+b(a+b)*a