MILITARY INSTITUTE OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COURSE TITLE: COMPILER, COURSE CODE: CSE-303, CT-1
TIME: 25 MINS; SPRING 2022
SET-A + B

		Marks
	Question 1: This question is regarded for the assessment of CO1.	
	Compiler is a language processing system which converts a source code to a target machine code. In this way of processing some other phases are also required along with 'compiler'. Briefly define and explain the role and purpose of each phase along with compiler.	6
	Question 2: This question is regarded for the assessment of CO2.	
	Demonstrate how source code is translated into a target code in a compiler using the input statement given below by showing all the phases of a compiler sequentially and the possible input and output of each phase: **Final= - num1 * 50 + num2 * 50**	14

all integer ⟹ Final = -num1 * 30 - num2 / 50

Q) "Compiler works in two steps whereas interpreter on 1 step". Define and explain the role and purpose of compiler and interpreter according to this statement. [6 marks]

MILITARY INSTITUTE OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COURSE TITLE: COMPILER, COURSE CODE: CSE-303, CT-2
DATE: 19-06-2022, TIME: 30 MINS: SPRING 2022
SET-1

Marks

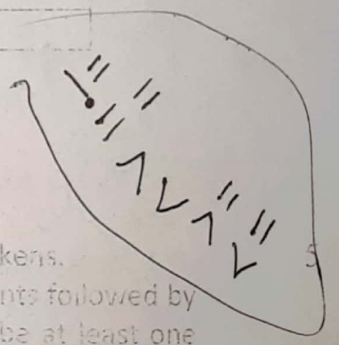
1. What should a diagram's code do if the character read at one point is not among the ones for which a following state has been specified? 7

2. Consider the following grammar for C language. Identify the lexemes that need to be converted to a token with appropriate token name and attribute value from this grammar in the following tabular format: 8

Lexemes	Token Name	Attribute Value

stmt-> do { *stmt* } while (*condition*) | ϵ
condition-> *term* relop *term* | ϵ
term-> id | number

do while



3. Write down the patterns and show the transition diagram for the following tokens. 5
Number: Consider of any non-zero number of digits (0-9). Allows decimal points followed by any non-zero number of digits. This fractional part is optional. There must be at least one digit after the decimal point. Exponential not considered. The number may contain positive and negative signs before it.
Correct: 6, 93, 6.2093, .99, 0.07, 57, -32.73 -7
Incorrect: 59.

MILITARY INSTITUTE OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COURSE TITLE: COMPILER, COURSE CODE: CSE-303, CT-2
DATE: 19-05-2022; TIME: 30 MINS; SPRING 2022
SET-A

Marks

1. While using two scheme buffers the forward pointer needs to check two conditions before incrementing once. Explain how this problem is solved using appropriate algorithms. 7

2 Consider the following grammar for C Language. Identify the lexemes that need to be converted to a token with appropriate token name and attribute value from this grammar in the following tabular format: 8

Lexemes	Token Name	Attribute Value

4 to 6

stmt -> for (*expr* ; *expr* ; *expr*) | ϵ
expr -> *term* assignmet_operator *term* | *term* increment_operator | *term* relop *term* | ϵ
term -> id | number

3 Write down the patterns and draw the transition diagram for the following tokens. 5
Number: Consider of any non-zero number of digits (0-9). Allows decimal points followed by any non-zero number of digits. This fractional part is optional. There must be at least one digit after the decimal point. Exponential not considered. Finally, the number cannot start with a 0.
Correct: 6, 90, 6.2098, .99, 57
Incorrect: 012, 0123.7, 59.

Set-B

17/12

CLASS TEST SCRIPT

2	0	2	0	1	4	0	6	0

YEAR DEPT CODE ROLL NO

COURSE CODE : CSE-303
COURSE TITLE : Compiler

MILITARY INSTITUTE OF SCIENCE AND TECHNOLOGY — BANGLADESH — TECHNOLOGY FOR ADVANCEMENT

Signature of Invigilator

LEVEL / YEAR : 3
TERM / SEMESTER : 1
DATE :

Q) 1) If an unidentified character is read with no following state, the lexical analyser reports a fail() command specifying that the specific transition was not able to identify the lexeme with the pattern. Although, this does not necessarily mean an error. In this situation, the diagram restarts and retracts where the first possible unidentification was found and moves on with another possible transition. The characters of the lexeme are read until a match a found [which by the diagram then performs a certain action for tokenisation].

If every possible diagram transition reports a fail() command, then the lexical analyser reports a true failure and tries to handle the error.

Q) 2)

From the given grammar, the lexemes that need to be identified are the keywords 'do', 'while', the identifiers or 'id', the numbers 'number', the seperators '{', '}', '(', ')', ';', and the lexemes for relational operators 'relop'.

Lexemes	Token Name	Attribute value
(	<(>	–
)	<)>	–
{	<{>	–
}	<}>	–
;	<;>	–
do	<do>	–
while	<while>	–
relop	<>>, <<>, <>=>, <<=>, <==>, <!=>	–
letter(letter+digit)*	<id>	Symbol table entry pointer for that specific identifier
digits(fraction)?(exponential)?	<number>	Symbol table entry pointer for that specific number.

Answer to the Question -3

Pattern :

digit → [0-9]

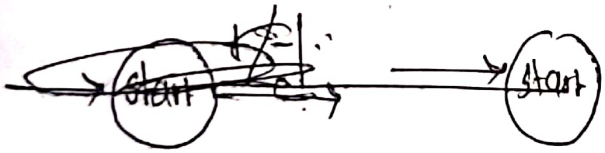
digits → digit$^+$

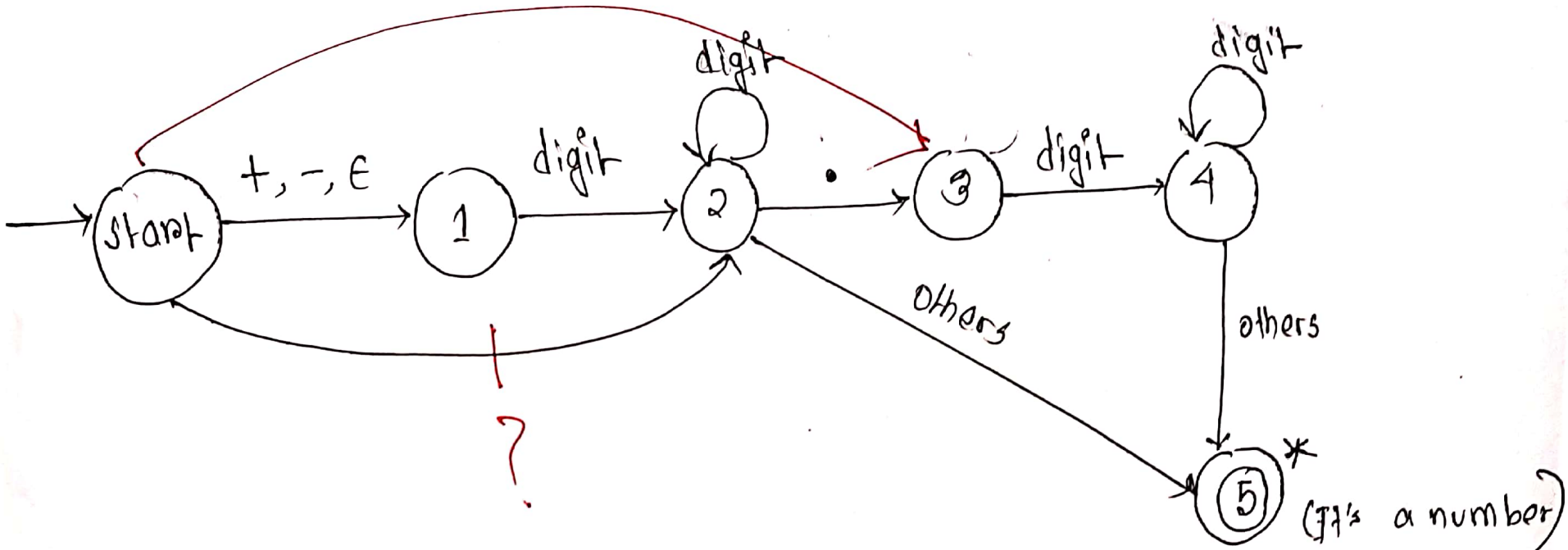
~~number → digits (.digits)~~

~~number → (+|-|ϵ) digits (.digits~~

number → (+|-|ϵ) digits (.digits)? | (.digits)

Transition diagram :-




start
+, −, ϵ
1
digit
2
digit
.
3
digit
4
digit
others
others
5
*
(It's a number)
?

MILITARY INSTITUTE OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COURSE TITLE: COMPILER, COURSE CODE: CSE-303, CT-2
DATE: 16-06-2022; TIME: 40 MINS; SPRING 2022
SET-A

This question is regarded for the assessment of CO3.

		Marks
1.	We have the following grammar for variable declaration in C: V ->T D ; D -> D , I | I T -> int | float | char I -> F E E -> = F | ∈ F -> id | number i. Make the grammar suitable for top-down parser. By performing the elimination of left recursion and then by performing left factoring. (if there is any). ii. Find the FIRST and FOLLOW of the variables iii. Construct the predictive parsing table using the above information.	4 + 14 + 8 =26
2.	Identify whether the following statement is syntactically correct by showing top-down parsing tree. int id, id;	4

MILITARY INSTITUTE OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COURSE TITLE: COMPILER, COURSE CODE: CSE-303, CT-2
DATE: 16-06-2022; TIME: 40 MINS; SPRING 2022
SET-B

This question is regarded for the assessment of CO3.

		Marks
1.	We have the following grammar for statement definition in C: S -> T id = E ; E -> E + E | E − E | E / E | (E) | T T -> id | num i. Make the grammar suitable for top-down parser. By performing the elimination of left recursion and then by performing left factoring. (if there is any). ii. Find the FIRST and FOLLOW of the variables iii. Construct the predictive parsing table using the above information.	5 + 10 + 10 =25
2.	Using the information from predictive parsing table Identify whether the above grammar is suitable for constructing predictive parser or not and explain your answer.	5

Shortcut Question:- Decide whether this grammar is LL(1) or not.

Military Institute of Science and Technology
Department of Computer Science and Engineering
Course Title: Compiler, Course Code: CSE-303, Class Test-3 Date: 18/7/2022
SET-A , Time: 25 Minutes

Consider the Following Syntax-Directed Definition.

	PRODUCTION	SEMANTIC RULES
1)	$E \rightarrow T\ E'$	$E.node = E'.syn$ $E'.inh = T.node$
2)	$E' \rightarrow +\ T\ E'_1$	$E'_1.inh = \textbf{new}\ Node('+', E'.inh, T.node)$ $E'.syn = E'_1.syn$
3)	$E' \rightarrow -\ T\ E'_1$	$E'_1.inh = \textbf{new}\ Node('-', E'.inh, T.node)$ $E'.syn = E'_1.syn$
4)	$E' \rightarrow \epsilon$	$E'.syn = E'.inh$
5)	$T \rightarrow (\ E\ )$	$T.node = E.node$
6)	$T \rightarrow \textbf{id}$	$T.node = \textbf{new}\ Leaf(\textbf{id}, \textbf{id}.entry)$
7)	$T \rightarrow \textbf{num}$	$T.node = \textbf{new}\ Leaf(\textbf{num}, \textbf{num}.val)$

1. Draw an annotated parse tree for the input "**a ÷ 7 - c + b**" using the given SDD. 5
2. Construct Syntax tree for the input **a + (c - 5) + b** with appropriate explanation. 10
3. Enlist the node construction steps in which the syntax tree is constructed in question-2. 5

Military Institute of Science and Technology
Department of Computer Science and Engineering
Course Title: Compiler, Course Code: CSE-303, Class Test: 3 Date: 18/7/2022
SET-B, Time: 25 Minutes,

der the Following Syntax-Directed Definition

	Production	Semantic Rules
1)	$E \rightarrow T\ E'$	$E.node = E'.syn$ $E'.inh = T.node$
2)	$E' \rightarrow + T\ E'_1$	$E'_1.inh = \text{new } Node('+', E'.inh, T.node)$ $E'.syn = E'_1.syn$
3)	$E' \rightarrow - T\ E'_1$	$E'_1.inh = \text{new } Node('-', E'.inh, T.node)$ $E'.syn = E'_1.syn$
4)	$E' \rightarrow \epsilon$	$E'.syn = E'.inh$
5)	$T \rightarrow (\ E\ )$	$T.node = E.node$
6)	$T \rightarrow \text{id}$	$T.node = \text{new } Leaf(\text{id}, \text{id}.entry)$
7)	$T \rightarrow \text{num}$	$T.node = \text{new } Leaf(\text{num}, \text{num}.val)$

Draw an annotated parse tree for the input "**a - 5 + b - d**" using the given SDD.

Construct Syntax tree for the input **5 + c - (a + 7)** with appropriate explanation.

Enlist the node construction steps in which the syntax tree is constructed in question-2.

Military Institute of Science and Technology
Department of Computer Science and Engineering
Course Title: Compiler, Course Code: CSE-303
Assignment in replace of Class Test-3

Following grammar generates binary numbers with a decimal point:

```
S ->   L . L | L
L ->   L B | B
B ->   0 | 1
```

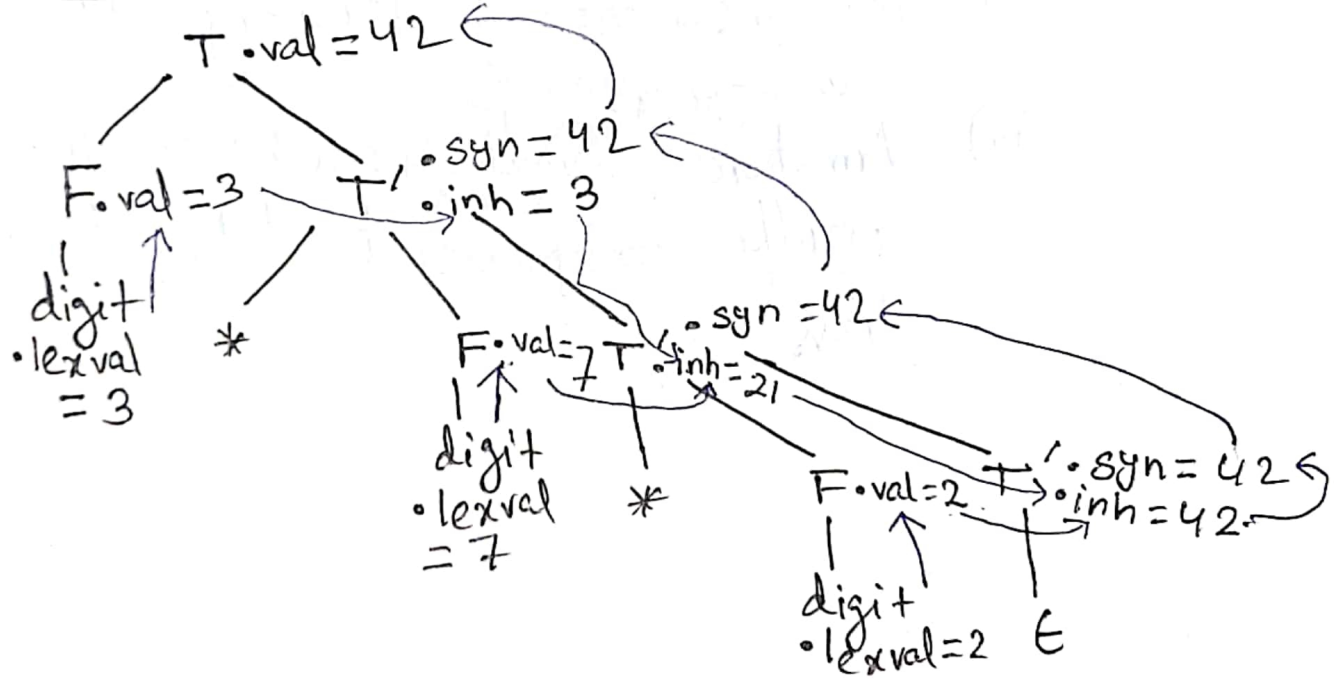
1. Draw a parse tree for each of the strings "10110.001" and "11011".

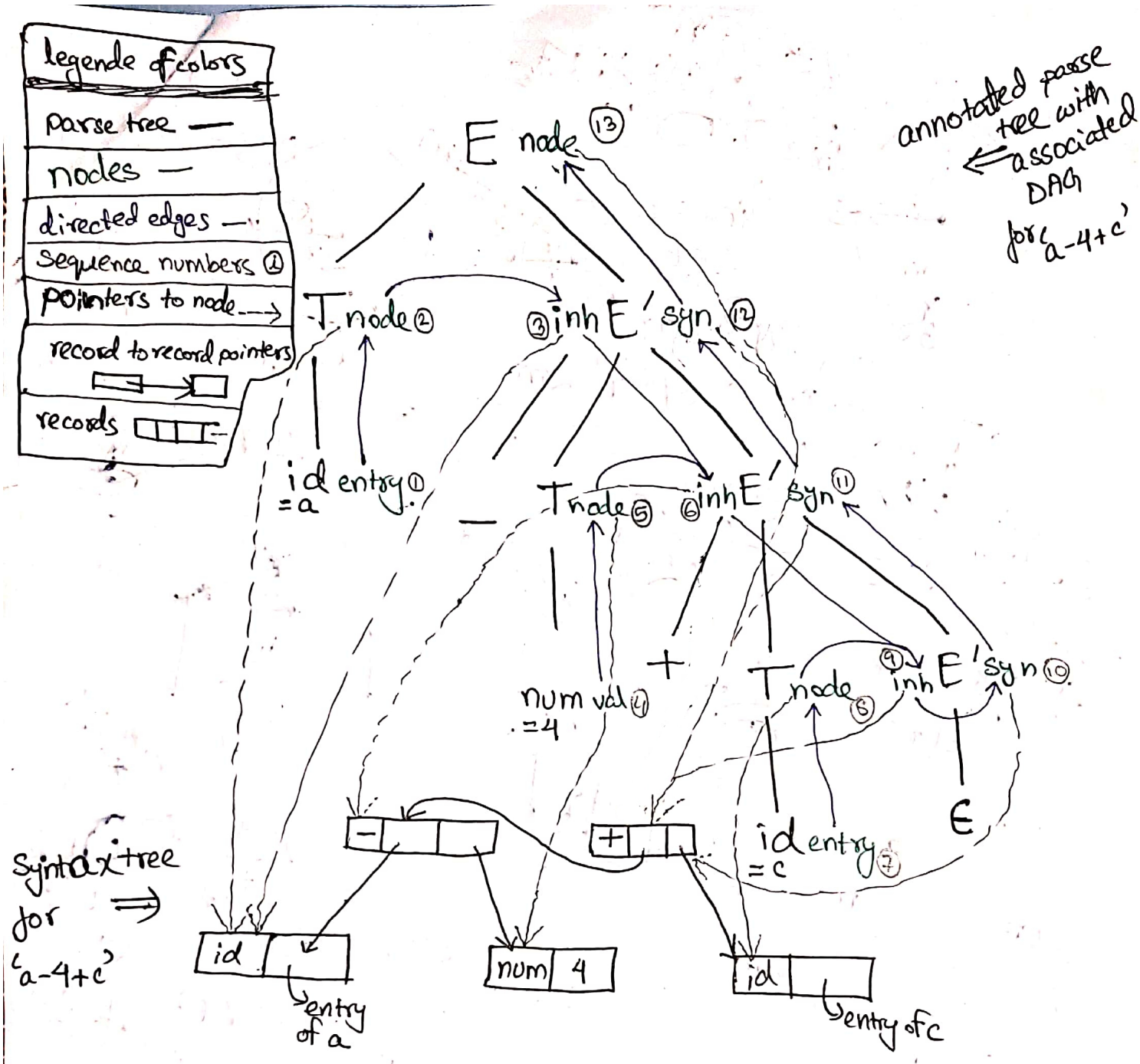
2. Design an L-attributed SDD to compute S.val, the decimal-number value of an input string. For example, the translation of string 101.101 should be the decimal number 5.625.
   Hint: use an inherited attribute L.side that tells which side of the decimal point a bit is on.

3. Design an S-attributed SDD for the grammar to do the same task as described in Question-1.

Production	Rules
$T \to F\ T'$	$T'.inh = F.val,\ T.val = T'.syn$
$T' \to *\ F\ T_1'$	$T_1'.inh = T'.inh * F.val,\ T'.syn = T_1'.syn$
$T' \to \epsilon$	$T'.syn = T'.inh$
$F \to digit$	$F.val = digit.lexval$

For, 3 * 7 * 2,




legende of colors
parse tree
nodes
directed edges
sequence numbers
pointers to node
record to record pointers
records
annotated parse tree with associated DAG for 'a-4+c'
E node 13
T node 2
3 inh E' syn 12
id entry 1
=a
T node 5
6 inh E' syn 11
num val 4
=4
+
T node 8
9 inh E' syn 10
id entry 7
=c
ε
Syntax tree for 'a-4+c'
id
entry of a
num 4
id
entry of c

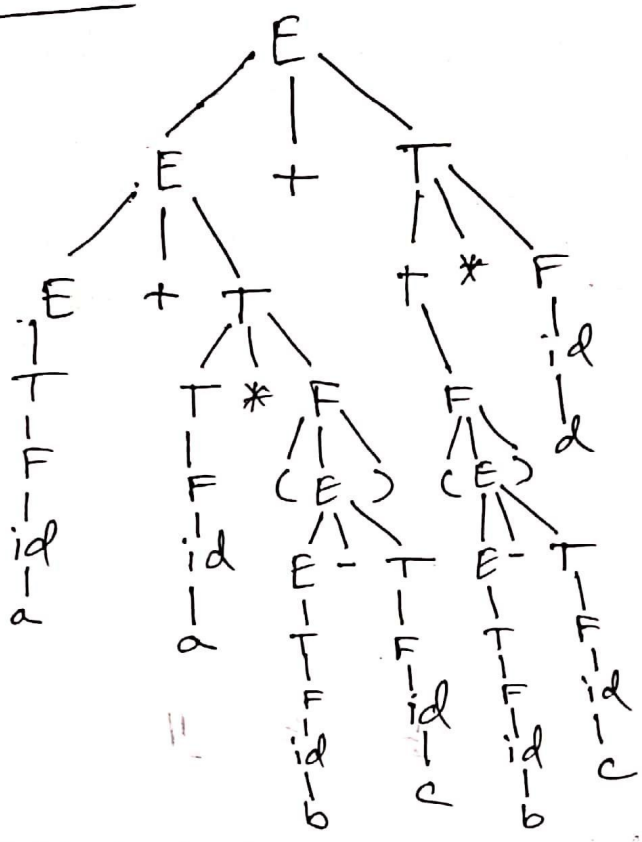
Example 6.1(Cont...)

The SDD of figure can construct either syntax trees or DAG's

PRODUCTION	SEMANTIC RULES
1) $E \rightarrow E_1 + T$	$E.node =$ **new** $Node('+', E_1.node, T.node)$
2) $E \rightarrow E_1 - T$	$E.node =$ **new** $Node('-', E_1.node, T.node)$
3) $E \rightarrow T$	$E.node = T.node$
4) $T \rightarrow T_1 * F$	$T.node =$ **new** $Node('*', T_1.node, F.node)$
5) $T \rightarrow T_1 / F$	$T.node =$ **new** $Node('/', T_1.node, F.node)$
6) $T \rightarrow F$	$T.node = F.node$
7) $F \rightarrow (E)$	$F.node = E.node$
8) $F \rightarrow$ **id**	$F.node =$ **new** $Leaf($**id**, **id**$.entry)$
9) $F \rightarrow$ **num**	$F.node =$ **new** $Leaf($**num**, **num**$.val)$

Q) Construct a DAG for a + a * (b-c) + (b-c) * d

Ans) Parse tree,



Steps in DAG construction,

P_1 = Leaf(id, entry a)

Not created { P_2 = Leaf(id, entry a) = P_1

P_3 = Leaf(id, entry b)

P_4 = Leaf(id, entry c)

P_5 = Node('-', P_3, P_4)

P_6 = Node('*', P_1, P_5)

P_7 = Node('+', P_1, P_6)

Not created { P_8 = Leaf(id, entry b) = P_3

Not created { P_9 = Leaf(id, entry c) = P_4

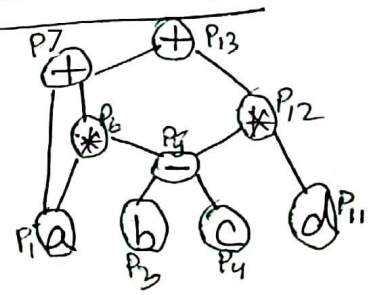
P_{10} = Node('-', P_3, P_4) = P_5 } Not created

P_{11} = Leaf(id, entry d)

P_{12} = Node('*', P_5, P_{11})

P_{13} = Node('+', P_7, P_{12})

∴ DAG is,



Source Code

```
x = 1;
y = x + 10;
while (x<y){
      x = x+1;
      if(x%2==1)
            y=y+1;
      y = y+2;
}
```

3 Address Code

```
100: t1 = 1
101: x = t1
102: t2 = x + 10
103: y = t2
104: if x < y goto 106
105: goto 11
106: t3 = x +1
107: x = t3
108: t4 = x%2
109: if t4 == 1 goto 111
110: goto 113
111: t5 = y+1
112: y = t5
113: t6 = y +2
114: y = t6
115: goto 104
116:
```

BANGLADESH UNIVERSITY OF PROFESSIONALS

Military Institute of Science and Technology

B.Sc. in Computer Science and Engineering, Term Final (Spring) Examination 2022: Aug 2022

Student Group: 72 < Earned Credit Hours ≤ 108

Subject: CSE–303: Compiler

Time: 3.00 hours **Full Marks: 180**

INSTRUCTIONS:

a. Use **SEPARATE** answer scripts for each section.
b. **Question-1** in **Section-A** and **Question-5** in **Section-B** are compulsory
c. Answer any other **TWO** questions out of **THREE** from each section.
d. Figures in the margin indicate full **marks**.
e. Assume reasonable data if necessary.
f. **Symbols** and **abbreviations** used have their usual meanings.

SECTION-A

Question 1 (Compulsory)

a. S → while (E) { S } | ε
E → T E'
E' → R T E' | ε
T → id | num
R → == | <= | >= | < | > | !=

i) Do you need to modify the above grammar before constructing a predictive top down parser? Justify your answer. If so, make the necessary changes and use the modified grammar for the following questions; otherwise use the given grammar. **6**

ii) Find FIRST and FOLLOW for each non-terminal of the above mentioned grammar **7**

iii) Build a predictive parsing table using the FIRST and FOLLOW set. **14**

iv) Using the information from predictive parsing table examine whether the above grammar is suitable for constructing predictive top-down parser or not and explain your answer. **3**

Question 2

a. Tokenize the given piece of source code using the patterns given in the Table 1. Find out if there are any lexical errors. **10**

Token	Attribute	Pattern
Keyword	Lexeme itself	Characters f,o,r Characters r,e,t,u,r,n Characters i,n,t Characters p,r,i,n,t
Bracket	Lexeme itself	(or) or { or }
Relational Operator	Lexeme itself	== or != or < or > o r <= or >=
Arithmetic Operator	Lexeme itself	+ or – or * or / or ++ or --
Assignment Operator	Lexeme itself	=
Literal	Lexeme itself	Characters surrounded by "and"
Identifier	Pointer to the symbol table	Letters followed by letters, digits or underscores ( )
Number	Pointer to the symbol table	Any numeric constants, allows decimal and exponential

Table 1: Token, their patterns and attribute value

```
1. int main()
2. {
3.  int i,fact=1,number=5;
4.  printf("Enter a number: ")$
5.     for(i=1;i<=number;i++){
6.       fact=fact*i
7.   }
8.  printf("Factorial of %d is: %d",number,
9.  return 0;
10.      }
```

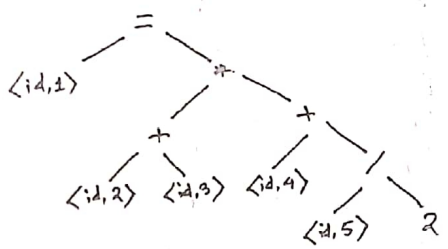
b. "Single buffer scheme" is enough while lexical analyzer scans characters from source file" - do you agree with this assumption? Justify your answer with relevant examples. 10

c. Construct the patterns and show the transition diagram that will match Date format. Day, Month and Year – all three parts must be present in the date format. The parts can be separated by either dot (.) or hyphen (-). Year must be written in 4 digit format. Needless to say that the numbers in the date should represent valid numbers. 10
Examples of some correct and incorrect date formats are given below:

Correct :
12-03-2022, 1-2-2000, 1.11.2050
01-10-2019, 01.05.1950, 1-02-2024
Incorrect:
36.9.2022, 3-14-2023, 12.12.22

Question 3

a. Suppose the semantic analyzer produces the following syntax tree for a specific statement. 10



i) Now, construct the three address code for this syntax tree.
ii) If the three address code is passed to the code optimizer, will there be any changes for the intermediate representations?
iii) Construct the assembly code of this intermediate representation.

b. Compiler is a language processing system which converts a source code to a target machine code. In this way of processing some other phases are also required along with compiler. Briefly define and explain the role and purpose of each phase along with compiler. 10

c. Suppose you have been tasked to design a compiler for different programming languages for same machine. Now choose between constructing a single pass compiler and a two pass compiler. Justify your response. 10

Question 4

a. What is indirect left recursion? Eliminate all kinds of recursion from the following grammar. 2+8

A -> B x y | x
B → C D

$C \rightarrow A \mid c$
$D \rightarrow d$

b. Explain with proper reason whether the following grammars are LL(1) or not? 5

i. $S \rightarrow a\,B \mid \varepsilon$, $B \rightarrow b\,C \mid \varepsilon$, $C \rightarrow C\,S \mid \varepsilon$
ii. $S \rightarrow a\,S\,A \mid \varepsilon$, $A \rightarrow c \mid \varepsilon$

c. During parsing, If the input pointer is pointing to the symbol a, the top of the stack symbol is X, with the entry in the parsing table being $X \rightarrow Y_1 Y_2 \ldots Y_k$, X will be popped and Y_1, $Y2 \ldots Y_k$ will be pushed with Y_k at the top. 5
Examine whether this statement is correct or not.

d. Summarize the significance of symbol table in different phases of compiler. 10

SECTION-B

Question 5 (Compulsory)

a. Illustrate the purpose of the Syntax Directed Definition shown in figure 5(a). Now construct an annotated parse tree for the string 1011011 and find the semantic value for the start variable in the parse tree. 10

b. Suppose you are assigned to design a Code Generator (Back end) of a compiler for a particular language . Now analyze the issues that need to be considered before starting the design. 8

c. Illustrate the three Address Code for the following arithmetic expression and translate into quadruples considering elements of array 'a' and 'b' as floating point types. 12

```
a=b[i]+c[i]+d;
i=i+1;
a[i]=b[i-1]*d-c[i+1]*d;
x=func(a[i]);
```

Question 6

a. Consider the Syntax Directed Definition (SDD) shown in figure 6(a). Now appraise the following questions: 20

i) Critically analyze the subclass of the given syntax directed definition.
ii) Construct annotated parse tree for the string '101101.'
iii) Identify the evaluation order of the annotated parse tree by constructing a dependency graph.
iv) Evaluate the value of the terminals and non terminals using the order identified in question 6 (iii).

b. Design a Syntax Directed Definition (SDD) to translate infix expression with + and * into equivalent expression in postfix expression where both + and * operations associate from left, and * takes precedence over +. You can assume necessary functions or 10

attributes as required to design the SDD. The designed SDD will translate a*(b+c)*d into abc+*d*.

Question 7

a. Write short note on each of the following terminologies with appropriate examples: 9
   i) Activation tree for procedure calling
   ii) Padding in storage allocation
   iii) Type Expression

b. Explain the way of calculating program cost. 6
   Calculate program cost for the target machine code shown in figure-7(b) considering c,i,a,p and x are name address.

c. Explain the elements of the activation record for functions. Construct downward growing stacks for the activation tree shown in figure 7(c) when the procedure q(7,9) is activated. 15

Question 8

a. Illustrate the algorithm for partitioning three-address instructions into basic blocks with appropriate examples. 10

b Applying the concept of intermediate representation in the three address codes, write three address codes for the program fragment of C language shown in figure-8(b). 5

c. For the pseudo code shown in figure-8(c), considering the instructions of a simple target machine, generate machine code using stack allocation for procedures main, set_root and set_value. 15
   Assume that the code for the procedure main, set_root and set_value start at the addresses 200, 400 and 600 respectively. Also assume that activation record size for procedures main, set_root and set_value are 40, 30 and 20 respectively. Further assume that action1, action2, action5 takes 16 bytes each, action3, action4, action6 instructions take 20 bytes each and the stack starts at the address 740. You may assume the registers and instructions as required for the target machine.

Production Rules	Semantic Rules
$B \rightarrow B_1 0$	$B.val = 2 \times B_1.val$
$B \rightarrow B_2 1$	$B.val = 2 \times B_1.val + 1$
$B \rightarrow 1$	$B.val = 1$

Figure-5(a): Syntax Directed Definition - (1)

Production	Semantic Rule
$B \rightarrow DB_1$	$B.pos := B_1.pos + 1$
	$B.val := B_1.val + D.val$
	$D.pow := B_1.pos$
$B \rightarrow D$	$B.pos := 1$
	$B.val := D.val$
	$D.pow := 0$
$D \rightarrow 0$	$D.val := 0$
$D \rightarrow 1$	$D.val := 2^{D.pow}$

Figure-6(a): Syntax Directed Definition - (2)

```
LD R0, c
LD R1, i
MUL R1, R1, 8
ST a(R1), R0
LD R0, p
LD R1, x
ST 0(R0), R1
```

Figure-7(b): Machine Code

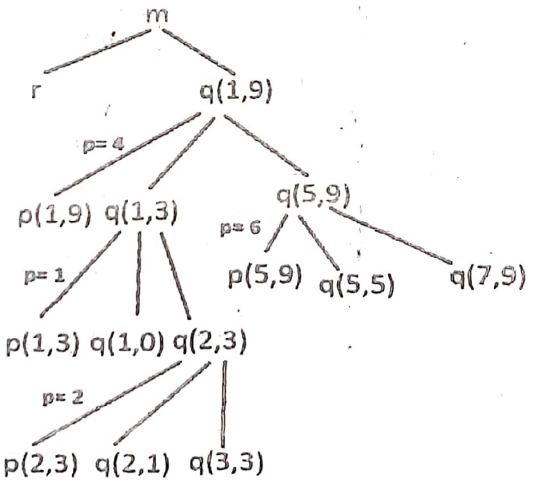


Figure-7(c): Activation Tree

```
int add(int a, int b){
  return a+b;
}

main(){
  int i = 5;int j = 3;int k=0;
  while(i>0){
    j = j+i+1;
    k=add(j, i);
    i=i-1;
  }
}
```

Figure-8(b): Program Fragment

```
// code for main
action1
action2
call set_root
action3
halt

// code for set_root
action5
action5
call set_value
return

// code for set_value
action6
action4
return
```

Figure-8(c): Pseudo-Code