18IT502

Hall Ticket Number:

III/IV B.Tech (Regular) DEGREE EXAMINATION

Feb, 2021 — **Information Technology**

Fifth Semester — **Automata & Compiler Design**

Time: Three Hours — **Maximum:** 50 Marks

Answer all questions from Part-A. (1X10= 10 Marks)

Answer ANY FOUR questions form Part-B. (4X10=40 Marks)

Part-A

1. Answer all questions (1X10=10 Marks)
   a) Define deterministic finite automata. CO1
   b) Define €-closure of a state. CO1
   c) What is the relation between $\Sigma^* = \Sigma^+$ CO1
   d) What is ambiguous grammar? CO2
   e) How many ways can PDA accepts the string? CO2
   f) Define compiler. CO3
   g) What are the difficulties with Top-Down parsing? CO3
   h) List the possible actions can make in shift-reduce parsing. CO3
   i) What are the different types of intermediate representations? CO4
   j) Define basic block? CO4

Part-B

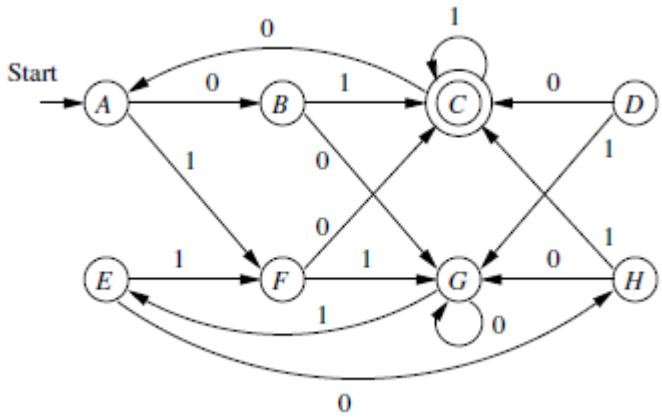
UNIT-I

2. a) Design DFA to accept the language L where L = {w/w has both an even number of 0's and even number of 1's}. CO1 5M

   b) Construct a DFA equivalent to the NFA given by M = ({p,q,r,s}, {0,1}, δ , p, {s}), where δ is defined in the following table CO1 5M

δ	0	1
→p	{p,q}	{p}
q	{r}	{r}
r	{s}	Φ
*s	{s}	{s}

3. a) Construct the minimum state equivalent DFA for the following CO1 5M



   b) Using pumping lemma prove L={ $0^m\ 1^n$ | m<n , n≥1 } is not regular. CO1 5M

UNIT-II

4. a) Consider the following grammar CO2 5M
   S → A1B
   A → 0A | €
   B → 0B | 1B | €
   Give leftmost and rightmost derivations of the following string **00101**

   b) Convert the grammar CO2 5M
   S →0S1 | A
   A → 1A0 | S | € to a PDA that accepts the same language by empty stack.

5.	a)	Construct a PDA that accepts the language L = { WCW^R / W$\in$ {a, b}* }.	CO2	5M
	b)	Convert the following grammar into CNF from G = ({S,A,B}, {a,b,c}, p, S) productions are S→ABa A→aab B→Ac	CO2	5M

UNIT-III

6.	a)	Explain the output of each phase of a compiler for the statement "Position = Initial + rate * 60.0"	CO3	5M
	b)	Explain the role of lexical analyzer.	CO3	5M
7.	a)	Test whether the grammar is LL(1) or not, and construct a predictive parsing table for E → E+T/ T, T → T*F/ F, F → (E)/ id	CO3	5M
	b)	Explain the stack implementation of Shift – reduce parser with an example.	CO3	5M

UNIT-IV

8.	a)	What is a three address code? Generate quadruples, triples and indirect triples for the expression w := a* - (b + c)	CO4	5M
	b)	What are the different storage allocation strategies available? Explain each in brief?	CO4	5M
9.	a)	Define basic block and flow graph. Write an algorithm to construct basic block.	CO4	5M
	b)	Write an algorithm for simple code generation and construct code sequence for the following expression: w:=(a-b)+(a-c)+(a-c) using code-generator algorithm.	CO4	5M

III/IV B.Tech (Regular) DEGREE EXAMINATION

Feb, 2021 **Information Technology**
Fifth Semester **Automata and Compiler Design (18IT502)**
Time: Three Hours **Maximum:** 50 Marks

Scheme of Evaluation & Answers

Part-A

1 Answer all questions (1X10=10 Marks)

a) Define deterministic finite automata.

A DFA is represented formally by a 5-tuple, (Q, Σ, δ, q0, F), consisting of

- a finite set of states Q
- a finite set of input symbols Σ
- a transition function δ: Q × Σ → Q
- an initial (or start) state q0 □ Q
- a set of states F distinguished as accepting (or final) states F □ Q.

b) Define €-closure of a state.

Epsilon (∈) – closure: Epsilon closure for a given state X is a set of states which can be reached from the states X with only (null) or ε moves including the state X itself.

c) What is the relation between $\Sigma^* = \Sigma^+$

Σ*(kleene star) is a unary operator on a set Σ of symbols or strings that gives an infinite collection of all possible strings of all possible lengths including λ (empty string). Σ+ is same as Σ* excluding empty string λ.

d) What is ambiguous grammar?

Ambiguous grammar: A CFG for which there are more than one left-most | right-most derivation trees or more than one derivation trees for a given string.

e) How many ways can PDA accepts the string?

Two ways. .i) Acceptance by final state. ii) Acceptance by empty stack.

f) Define compiler.

A compiler is a computer program that translates computer code written in one programming language (the source language) into another language (the target language).

g) What are the difficulties with Top-Down parsing?

- Backtracking.
- · Left recursion.
- · Left factoring.
- · Ambiguity.

h) List the possible actions can make in shift-reduce parsing.

1) Shift 2) Reduce

3) Accept 4) Error

i) What are the different types of intermediate representations?

Syntax tree

Postfix notation (or) Reverse polish notation

Intermediate code

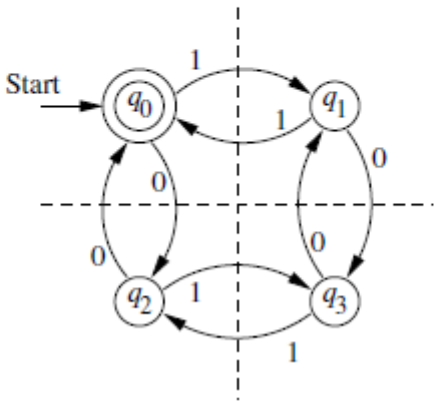
j) Define basic block?

Basic Block is a straight line code sequence which has no branches in and out branches except to the entry and at the end respectively. Basic Block is a set of statements which always executes one after other, in a sequence.

Part-B
UNIT-I

2 a) Design DFA to accept the language L where L = {w/w has both an even number of 0's and even number of 1's}. **5M**

State q_0 is both the start state and the lone accepting state. It is the start state, because before reading any inputs, the numbers of 0's and 1's seen so far are both zero, and zero is even. It is the only accepting state, because it describes exactly the condition for a sequence of 0's and 1's to be in language L.



The DFA for language L is

$$A = (\{q_0, q_1, q_2, q_3\}, \{0,1\}, \delta, q_0, \{q_0\})$$

Where the transition function δ is described by the transition diagram

	0	1
$* \rightarrow q_0$	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

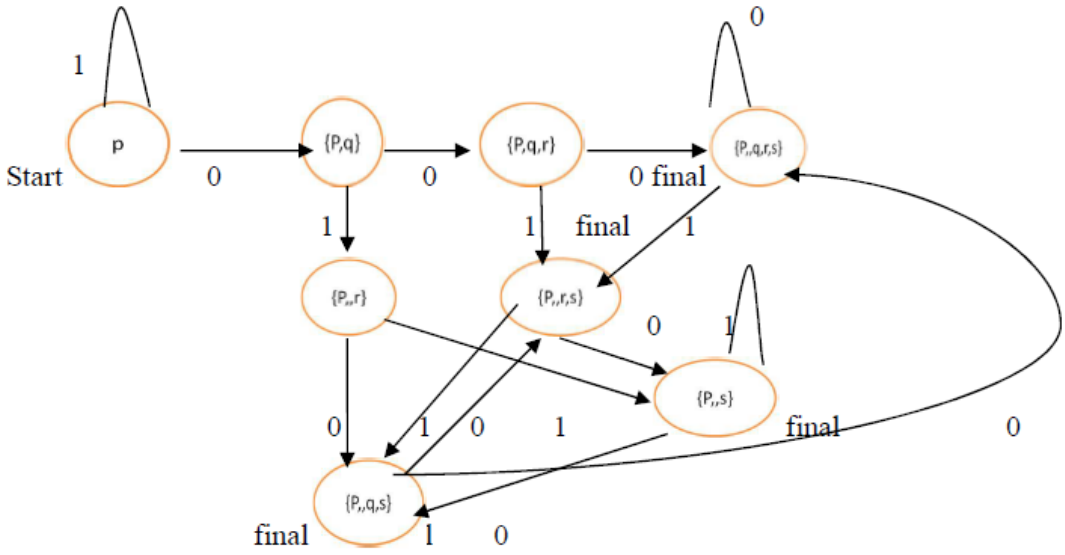
2 b) Construct a DFA equivalent to the NFA given by M = ({p,q,r,s}, {0,1}, δ, p, {s}), where δ is defined in the following table **5M**

δ	0	1
→p	{p,q}	{p}
q	{r}	{r}
r	{s}	Φ
*s	{s}	{s}

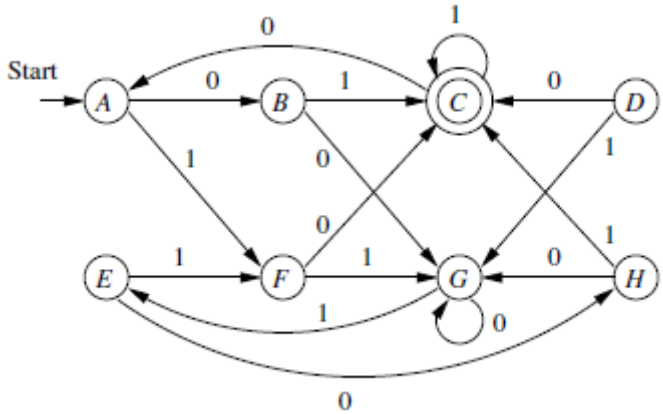
The following table represents NFA to DFA conversion :

δ	0	1
->p	{p,q}	{p}
{p,q}	{p,q,r}	{p,r}
{p,q,r}	{p,q,r,s}	{p,r,s}
*{p,q,r,s}	{p,q,r,s}	{p,r,s}
{p,r}	{p,q,s}	{p,s}
*{p,r,s}	{p,q,s}	{p,s}
*{p,q,s}	{p,q,r,s}	{p,r,s}
*{p,s}	{p,q,s}	{p,s}

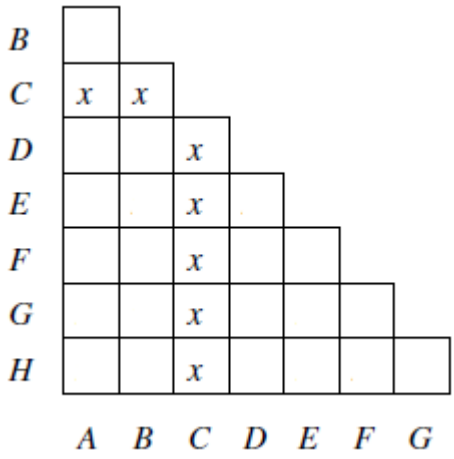
Transition diagram



3 a) Construct the minimum state equivalent DFA for the following **5M**



For the basis, since C is the only accepting state, we put **x** in each pair that involves C.



Now, we know some distinguishable pairs, we can discover others.

Consider pair (A,B): δ(A,0) →B, δ(B,0) →G ⎫ X ⎧ δ(A,1) →F, δ(B,1) →C ⎭ → Put X in (A,B)

Next Pair (A,D): δ(A,0) →B, δ(D,0) →C → Put X in (A,D) Pair

Next pair (A,E): δ(A,0) →B, δ(E,0) →H X δ(A,1) →F, δ(E,1) →F X

Next pair (A,F): δ(A,0) →B, δ(F,0) →C → Put X in (A,F) Pair

Next Pair (A,G): δ(A,0) →B, δ(G,0) →G X δ(A,1) →F, δ(G,1) →E X

Next Pair (AH): δ(A,0) →B, δ(H,0) →G X δ(A,1) →F, δ(H,1) →C →Put X in (A,H)

Continue like this for remaining pairs also....

i.e (B,D), (B,E), (B,F), (B,G), (B,H), (D,E), (D,F), (D,G), (D,H), (E,F), (E,G), (E,H), (F,G), (F,H), and (G,H).

Observe the above table (A,E), (A,G), (B,H), (D,F), and (E,G) pairs are empty.

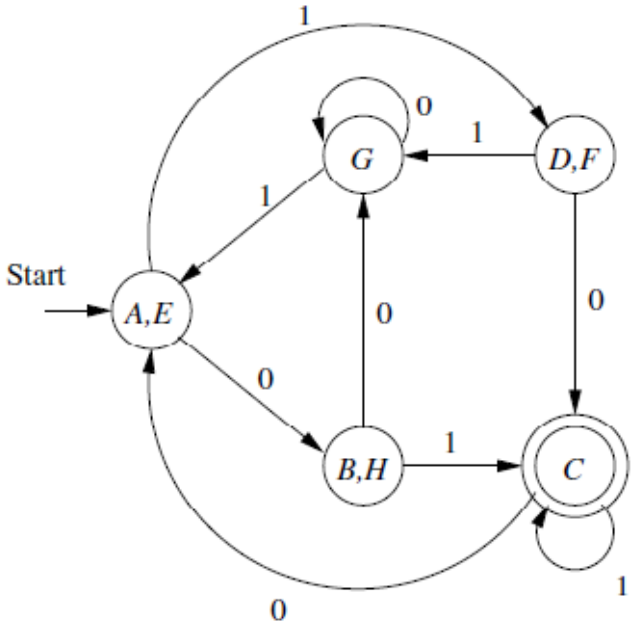
consider pair (A,G) with 'X', δ(A,0) →B, δ(G,0) →G →(B,G) distinguishable pair or filled So, pair (A,G) also distinguishable pair.

Consider pair (E,G) with 'X', δ(E,0) →H, δ(G,0) →G →(H,G) distinguishable pair or filled So, pair (E,G) also distinguishable pair.

B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x
	A	B	C	D	E	F	G

Hence, (A,G), (B,H), and (D,F) are equivalent states.

So, Minimum State DFA is:



3 b) Using pumping lemma prove L={ 0^m 1^n | m<n , n≥1 } is not regular. **5M**

Let L be a regular language. Then there exists a constant 'c' such that for every string w in L such that |w| ≥ c.

We can break the string w into three parts, w = xyz, such that

- |y| > 0
- |xy| ≤ c
- For all k ≥ 0, the string xykz is also in L.

Consider the string from L

w= 0000111= xyz

x=000; y=01; z=11

w= xy^iz= 000 $(01)^i$ 11

If i= 2 then 000010100 not belongs to L

Hence, the given language is not regular.

UNIT-II

4 a) Consider the following grammar **5M**

S → A1B
A → 0A | €
B → 0B | 1B | €
Give leftmost and rightmost derivations of the following string **00101**

LMD: for string 00101	RMD: for string 00101
S →A1B	S →A1B
→ 0A1B	→A10B
→00A1B	→A101B
→00€1B	→A101€
→0010B	→0A101
→00101B	→00A101
→00101€	→00€101
→00101	→00101

4 b) Convert the grammar **5M**
S →0S1 | A
A → 1A0 | S | € to a PDA that accepts the same language by empty stack.

1. For each variable A,

$$\delta(q, \epsilon, A) = \{(q, \beta) \mid A \to \beta \text{ is a production of } G\}$$

2. For each terminal a, $\delta(q, a, a) = \{(q, \epsilon)\}$.

Variables in the given grammar are: S and A
δ (q, €, S) = { (q,0S1) }
δ (q, €, S) = { (q, A) }
δ (q, €, A) = { (q,1A0) }
δ (q, €, A) = { (q,S) }
δ (q, €, A) = { (q, €) }

Terminals in the given grammar are: 0 and 1
δ (q, 0,0) = { (q, €) }
δ (q, 1,1) = { (q, €) }

5 a) Construct a PDA that accepts the language L = { WCW^R / W€ {a, b}* }. **5M**

Some string will come followed by one 'c', followed by reverse of the string before 'c'.

So we get to know that 'c' will work as an alarm to starting poping STACK.

So we will pop every 'a' with 'a' and every 'b' with 'b'.

For every two a's and b's push them into STACK

When 'c' comes do nothing.

Starting poping STACK: 'a' for 'a' and 'b' for 'b'.

$\delta(q0, a, Z) = (q0, aZ)$

$\delta(q0, a, a) = (q0, aa)$

$\delta(q0, b, Z) = (q0, bZ)$

$\delta(q0, b, b) = (q0, bb)$

$\delta(q0, a, b) = (q0, ab)$

$\delta(q0, b, a) = (q0, ba)$

// this is decision step

$\delta(q0, c, a) = (q1, a)$

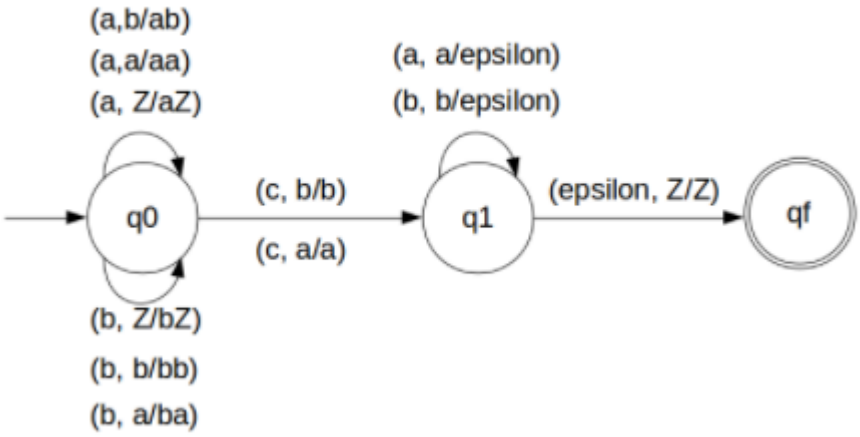
$\delta(q0, c, b) = (q1, b)$

$\delta(q1, b, b) = (q1, \varepsilon)$

$\delta(q1, a, a) = (q1, \varepsilon)$

$\delta(q1, \varepsilon, Z) = (qf, Z)$

We have designed the PDA for the problem:



5 b) Convert the following grammar into CNF from G = ({S,A,B}, {a,b,c}, p, S) productions are **5M**

S→ABa
A→aab
B→Ac

A CFG is in Chomsky Normal Form if the Productions are in the following forms –

Non terminal -> non terminal . non terminal (or)

Non terminal -> terminal

Example: Conversion to CNF

i)Removal of null productions

There are no nll productions

ii) Remove unit productions

There are no unit productions

iii) Conversion of each production

Replace each production A → B1…Bn where n > 2 with A → B1C where C → B2 …Bn.

Repeat this step for all productions having two or more symbols in the right side.

S->ABa can be written as S→AC C→BD D→a

A->aab can be written as A→DE E→DF F→b

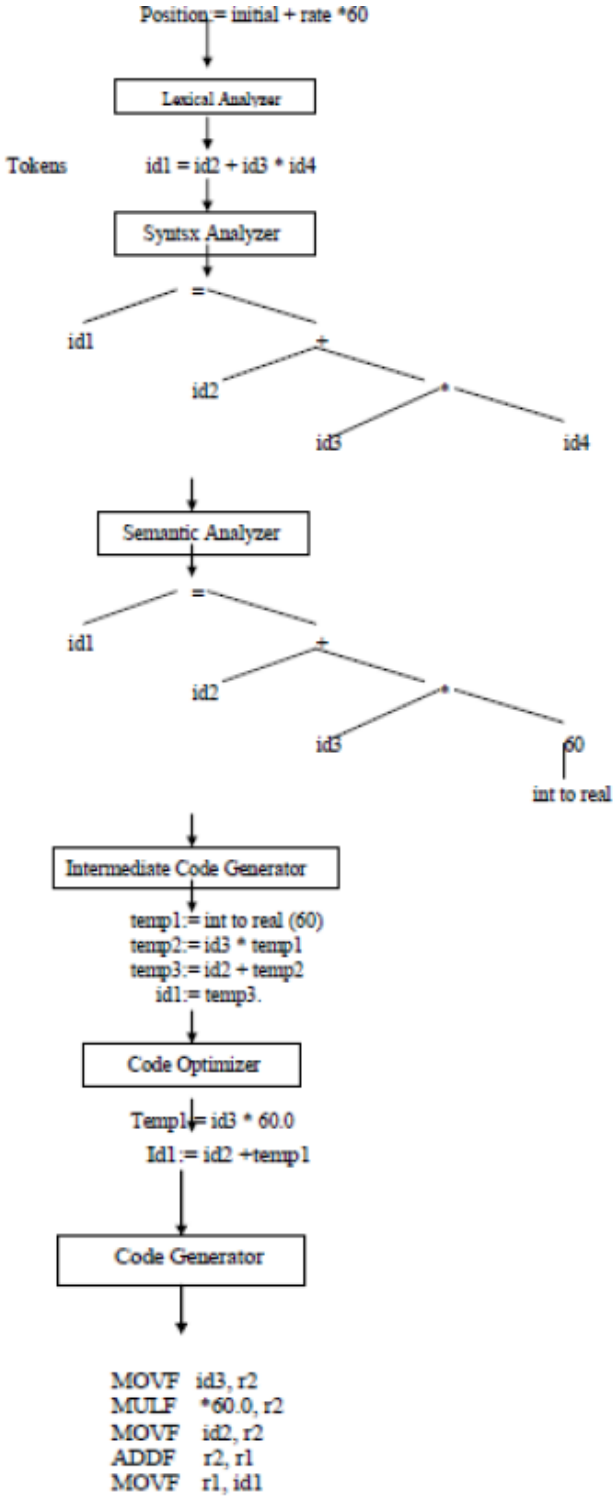
B->Ac can be written as B→AG G→c

The final CFG in Chomsky normal form is

S→AC C→BD D→a A→DE E→DF F→b B→AG G→c

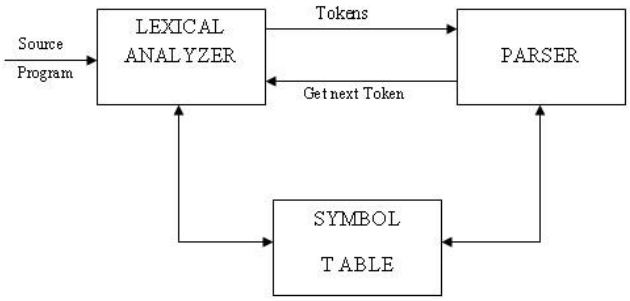
UNIT-III

6 a) Explain the output of each phase of a compiler for the statement "Position = Initial + rate * 60.0" **5M**



6 b) Explain the role of lexical analyzer. **5M**

The LA is the first phase of a compiler. Its main task is to read the input character and produce as output a sequence of tokens that the parser uses for syntax analysis.



Upon receiving a 'get next token' command form the parser, the lexical analyzer reads the input character until it can identify the next token. The LA return to the parser representation for the token it has found. The representation will be an integer code, if the token is a simple construct such as parenthesis, comma or colon.

LA may also perform certain secondary tasks as the user interface. One such task is striping out from the source program the commands and white spaces in the form of blank, tab and new line characters. Another is correlating error message from the compiler with the source program.

7 a) Test whether the grammar is LL(1) or not, and construct a predictive parsing table for E → E+T/ T, T → T*F/ F, F → (E)/ id **5M**

First eliminate the left recursion for E as

E → TE'
E' → +TE' | ε

Then eliminate for T as

T → FT'
T'→ *FT' | ε

Thus the obtained grammar after eliminating left recursion is

E → TE'
E' → +TE' | ε
T → FT'
T' → *FT' | ε
F → (E) | id

First( ) :

FIRST(E) = { (, id}
FIRST(E') ={+ , ε }
FIRST(T) = { (, id}
FIRST(T') = {*, ε }
FIRST(F) = { (, id }

Follow( ):

FOLLOW(E) = { $,) }
FOLLOW(E') = { $,) }
FOLLOW(T) = { +, $,) }
FOLLOW(T') = { +, $,) }
FOLLOW(F) = {+, * , $,) }

Predictive parsing table :

NON-TERMINAL	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E'→ ε
T	T → FT'			T → FT'		
T'		T'→ ε	T'→ *FT'		T' → ε	T' → ε
F	F → id			F → (E)		

Theparsing table has no multyply defined entries.
Hence, the given grammar is LL(1) grammar.

7 b) Explain the stack implementation of Shift – reduce parser with an example. 5M

Actions in shift-reduce parser:

- shift – The next input symbol is shifted onto the top of the stack
- reduce – The parser replaces the handle within a stack with a non-terminal.
- accept – The parser announces successful completion of parsing.
- error – The parser discovers that a syntax error has occurred and calls an error recovery routine.

Stack implementation of shift-reduce parsing :

Stack	Input	Action
$	id_1+id_2*id_3 $	shift
$ id_1	+id_2*id_3 $	reduce by E→id
$ E	+id_2*id_3 $	shift
$ E+	id_2*id_3 $	shift
$ E+id_2	*id_3 $	reduce by E→id
$ E+E	*id_3 $	shift
$ E+E*	id3 $	shift
$ E+E*id3	$	reduce by E→id
$ E+E*E	$	reduce by E→ E *E
$ E+E	$	reduce by E→ E+E
$ E	$	accept

UNIT-IV

8 a) What is a three address code? Generate quadruples, triples and indirect triples for the expression w := a* - (b + c) 5M

Three address code is a type of intermediate code which is easy to generate and can be easily converted to machine code. It makes use of at most three addresses and one operator to represent an expression and the value computed at each instruction is stored in temporary variable generated by compiler.

Three address code for the given expression: T1:= b+c

T2:=uminus T1

T3:=a*T2

w:=T3

Quaruples:

OP	arg1	arg2	Result
+	b	c	T1
uminus	T1	-	T2
*	a	T2	T3
assign	T3	-	w

Triples:

	OP	arg1	arg2
(0)	+	b	c
(1)	uminus	T1	-
(2)	*	a	(1)
(3)	assign	w	(2)

8 b) What are the different storage allocation strategies available? Explain each in brief? 5M

The different ways to allocate memory are:

1. Static storage allocation 2. Stack storage allocation 3.Heap storage allocation

Static storage allocation

- In static allocation, names are bound to storage locations.
- If memory is created at compile time then the memory will be created in static area and only once.
- Static allocation supports the dynamic data structure that means memory is created only at compile time and deallocated after program completion.
- The drawback with static storage allocation is that the size and position of data objects should be known at compile time.
- Another drawback is restriction of the recursion procedure.

Stack Storage Allocation

- In static storage allocation, storage is organized as a stack.
- An activation record is pushed into the stack when activation begins and it is popped when the activation end.
- Activation record contains the locals so that they are bound to fresh storage in each activation record. The value of locals is deleted when the activation ends.
- It works on the basis of last-in-first-out (LIFO) and this allocation supports the recursion process.

Heap Storage Allocation

- Heap allocation is the most flexible allocation scheme.
- Allocation and deallocation of memory can be done at any time and at any place depending upon the user's requirement.
- Heap allocation is used to allocate memory to the variables dynamically and when the variables are no more used then claim it back.
- Heap storage allocation supports the recursion process.

9 a) Define basic block and flow graph. Write an algorithm to construct basic block. **5M**

Basic block is a set of statements that always executes in a sequence one after the other.

Flow Graph is a directed **graph** with **flow** control information added to the **basic blocks**.

Basic Block Construction:

Algorithm: Partition into basic blocks

Input: A sequence of three-address statements

Output: A list of basic blocks with each three-address statement in exactly one block

Method:

1. We first determine the set of *leaders*, the first statements of basic blocks. The rules we use are of the following:
    a. The first statement is a leader.
    b. Any statement that is the target of a conditional or unconditional goto is a leader.
    c. Any statement that immediately follows a goto or conditional goto statement is a leader.
2. For each leader, its basic block consists of the leader and all statements up to but not including the next leader or the end of the program.

9 b) Write an algorithm for simple code generation and construct code sequence for the following expression: w:=(a-b)+(a-c)+(a-c) using code-generator algorithm. **5M**

For an instruction x = y OP z, the code generator may perform the following actions. Let us assume that L is the location (preferably register) where the output of y OP z is to be saved:

- Call function getReg, to decide the location of L.
- Determine the present location (register or memory) of **y** by consulting the Address Descriptor of **y**. If **y** is not presently in register **L**, then generate the following instruction to copy the value of **y** to **L**:

  MOV y', L where **y'** represents the copied value of **y**.
- Determine the present location of **z** using the same method used in step 2 for **y** and generate the following instruction:

  OP z', L where **z'** represents the copied value of **z**.
- Now L contains the value of y OP z, that is intended to be assigned to **x**. So, if L is a register, update its descriptor to indicate that it contains the value of **x**. Update the descriptor of **x** to indicate that it is stored at location **L**.
- If y and z has no further use, they can be given back to the system.

The expression W= (A- B) + (A – C) + (A – C) might be translated in to the following three address code:

t := a – b

u := a – c

v := t + u

d := v + u with d live at the end.

Code sequence for the example is:

Statements	Code Generated	Register descriptor	Address descriptor
		Register empty	
t : = a - b	MOV a, R_0 SUB b, R0	R_0 contains t	t in R_0
u : = a - c	MOV a , R1 SUB c , R1	R_0 contains t R1 contains u	t in R_0 u in R1
v : = t + u	ADD R_1, R_0	R_0 contains v R_1 contains u	u in R_1 v in R_0
d : = v + u	ADD R_1, R_0 MOV R_0, d	R_0 contains d	d in R_0 d in R_0 and memory

Scheme prepared by

Mr. G.Prasad, Asst.Professor.

Department of I.T. **Signature of the HOD, IT DEPT.**

Paper Evaluators:

S.No	Name of the College	Name of the Examiner	Signature