Sri Indu
College of Engineering & Technology
UGC Autonomous Institution
Recognized under 2(f) & 12(B) of UGC Act 1956,
NAAC, Approved by AICTE &
Permanently Affiliated to JNTUH
Estd.2001

NAAC
NATIONAL ASSESSMENT AND ACCREDITATION COUNCIL

COMPILER CONSTRUCTION LAB MANUAL

III Year IT –Semester II

DEPARTMENTOFINFORMATION TECHNOLOGY

ACADEMIC YEAR 2022-23

DEPARTMENT OF INFORMATION TECHNOLOGY

HANDOUT- INDEX

S. No	Contents
1	Vision, Mission, PEOs, POs, PSOs & COs
2	Institution Academic Calendar
3	Department Academic Calendar
4	Syllabus Copy
i)	Index
ii)	Programs
iii)	Model Paper For Lab Internal-1
iv)	Model Paper For Lab Internal-2

SRI INDU COLLEGE OF ENGINEERING & TECHNOLOGY

B. TECH –INFORMATION TECHNOLOGY

INSTITUTION VISION

To be a premier Institution in Engineering & Technology and Management with competency, values and social consciousness.

INSTITUTION MISSION

IM1 Provide high quality academic programs, training activities and research facilities.

IM2 Promote Continuous Industry-Institute Interaction for Employability,Entrepreneurship, Leadership and Research aptitude among stakeholders.

IM3 Contribute to the Economical and technological development of the region, state and nation.

DEPARTMENT VISION

To be a recognized knowledge center in the field of Information Technology with self -motivated, employable engineers to society.

DEPARTMENT MISSION

The Department has following Missions:

DM1 To offer high quality student centric education in Information Technology.

DM2 To provide a conducive environment towards innovation and skills.

DM3 To involve in activities that provide social and professional solutions.

DM4 To impart training on emerging technologies namely cloud computing and IOT with involvement of stake holders.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO 1: Higher Studies: Graduates with an ability to apply knowledge of Basic sciences and programming skills in their career and higher education.

PEO 2: Lifelong Learning: Graduates with an ability to adopt new technologies for ever changing IT industry needs through Self-Study, Critical thinking and Problem solving skills.

PEO 3: Professional skills: Graduates will be ready to work in projects related to complex problems involving multi-disciplinary projects with effective analytical skills.

PEO 4: Engineering Citizenship: Graduates with an ability to communicate well and exhibit social, technical and ethical responsibility in process or product.

PROGRAM OUTCOMES (POs) & PROGRAM SPECIFIC OUTCOMES (PSOs)

PO	Description
PO 1	**Engineering Knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, andan engineering specialization to the solution of complex engineering problems.
PO 2	**Problem Analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO 3	**Design / development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO 4	**Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO 5	**Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO 6	**The engineer and Society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO 7	**Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO 8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice
PO 9	**Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO 10	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO 11	**Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO 12	**Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological Change.
Program Specific Outcomes	
PSO 1	**Software Development:** To apply the knowledge of Software Engineering, Data Communication, Web Technology and Operating Systems for building IOT and Cloud Computing applications.
PSO 2	**Industrial Skills Ability:** Design, develop and test software systems for world-wide network of computers to provide solutions to real world problems.
PSO 3	**Project implementation:** Analyze and recommend the appropriate IT Infrastructure required for the implementation of a project.

SRI INDU COLLEGE OF ENGINEERING & TECHNOLOGY
(An Autonomous Institution under UGC, New Delhi)
Recognized under 2(f) and 12(B) of UGC Act 1956
NBA & NAAC Accredited, Approved by AICTE and Permanently affiliated to JNTUH
Sheriguda (V), Ibrahimpatnam, R.R.Dist, Hyderabad - 501 510

D4

BR-20

Lr.No.SICET/AUTO/DAE/III B.Tech Academic Calendar/307/2022 **Dt: 03.08.2022**

Dr.G. SURESH,
Principal,

To,
All the HODs.

III B.TECH I SEM & II SEM ACADEMIC CALENDAR
ACADEMIC YEAR : 2022-23

Sir,

Sub: SICET (Autonomous) - Academic & Evaluation - Academic Calendar for **B.Tech – 3rd Year** - For the academic year **2022-23** – Reg.

***

The approved Academic Calendar for **B.Tech – 3rd Year (I & II Sem)** for the academic year **2022-23** is given below:

Academic Calendar for B.Tech – 3rd Year Students
(2020 - 21 Batch), BR-20 Regulation.

I - Semester

Commencement of I Semester class work	**25.08.2022 (Thursday)**	
I Spell of Instructions. (Including CRT and Dussehra Holidays).	25.08.2022	02.11.2022 - 10 Weeks
Dussehra Holidays.	03.10.2022	08.10.2022 - 1 Week
I Mid Examinations for III B.Tech I Sem Students.	03.11.2022	05.11.2022 - 3 Days
II Spell of Instructions.	07.11.2022	31.12.2022 - 8 Weeks
II Mid Examinations for III B.Tech I Sem Students.	02.01.2023	04.01.2023 - 3 Days
Preparation Holidays, Practical Lab Examinations and Remedial Mid Test (RMT).	05.01.2023	18.01.2023 - 2 Weeks
Sankranti Holidays	13.01.2023	16.01.2023 - 4 Days
III B.Tech I Semester End Examinations (Main) and Supplementary Examinations.	19.01.2023	01.02.2023 - 2 Weeks
Commencement of class work of III B.Tech II Semester - 02.02.2023 (Thursday)		

II - Semester

Commencement of II Semester class work	**02.02.2023 (Thursday)**	
I Spell of Instructions.	02.02.2023	29.03.2023 - 8 Weeks
I Mid Examinations for III B.Tech II Sem Students.	31.03.2023	03.04.2023 - 3 Days
II Spell of Instructions (Including Summer Vacation).	04.04.2023	12.06.2023 - 10 Weeks
Summer Vacation.	15.05.2023	27.05.2023 - 2 Weeks
II Mid Examinations for III B.Tech II Sem Students.	13.06.2023	15.06.2023 - 3 Days
Preparation Holidays, Practical Lab Examinations and Remedial Mid Test (RMT).	16.06.2023	25.06.2023 - 10 Days
III B.Tech II Semester End Examinations (Main) and Supplementary Examinations.	26.06.2023	08.07.2023 - 2 Weeks
Commencement of class work of IV B.Tech I Semester - 10.07.2023 (Monday)		

ACE **CE** **DIRECTOR** **PRINCIPAL**

Copy to DAE.
Copy to all the Heads of the Depts.

CONTROLLER OF EXAMINATIONS
Sri Indu College of Engineering & Technology
(An Autonomous Institution under JNTUH)
Sheriguda (V), Ibrahimpatnam, R.R. Dist-501510.

DIRECTOR
(Academic Audit)
Sri Indu College of Engineering & Technology
Sheriguda, IBP, R.R. Dist-501510.

PRINCIPAL
Sri Indu College of Engineering & Technology
(An Autonomous Institution under JNTUH)
Sheriguda (V), Ibrahimpatnam, R.R.Dist.-501510.

SRI INDU COLLEGE OF ENGINEERING AND TECHNOLOGY (AUTONOMOUS)

DEPARTMENT OF INFORMATION TECHNOLOGY

DEPARTMENT CALENDAR – 2022-2023 (SEMESTER-II)

DAYS												JULY'23
SUNDAY								MAY'23				
MONDAY							1					
TUESDAY		FEBRAURY'23		MARCH'23			2					
WEDNESDAY	1		1				3			JUNE'23		
THURSDAY	2		2				4		1	III-II(CRT CLASSES) END EXAM (IV-II)		
FRIDAY	3		3			APRIL'23	5		2	III-II(CRT CLASSES) END EXAM (IV-II)		
SATURDAY	4		4		1		6		3	III-II(CRT CLASSES)	1	
SUNDAY	5	HOLIDAY	5	HOLIDAY	2	HOLIDAY GANDHI JAYANTHI	7	HOLIDAY	4	HOLIDAY	2	HOLIDAY
MONDAY	6		6		3	III-II MID-I EXAM(ADA&STM)	8		5	National Environment Day END EXAM (IV-II)	3	III-II END EXAM
TUESDAY	7		7	HOLI	4	III-II MID-I EXAM(ITE)	9	IV-II PROJECT REVIEW-2	6		4	
WEDNESDAY	8		8		5	JAGJIVAN JAYANTHI	10		7		5	III-II END EXAM
THURSDAY	9		9		6	III-II LAB INTERNAL (CC &ML)	11		8	III-II LAB INTERNAL (STM	6	
FRIDAY	10		10		7	GOOD FRIDAY	12	ANNUAL DAY	9	III-II LAB INTERNAL (CC)	7	
SATURDAY	11		11		8		13	SUMMER VACATION	10	III-II LAB INTERNAL (ML	8	
SUNDAY	12	HOLIDAY	12	HOLIDAY	9	HOLIDAY	14	HOLIDAY	11	HOLIDAY	9	HOLIDAY
MONDAY	13		13		10	III-II LAB INTERNAL (STM)	15	SUMMER VACATION	12		10	
TUESDAY	14		14		11		16	SUMMER VACATION	13		11	
WEDNESDAY	15		15		12	IV-II PROJECT REVIEW-1	17	SUMMER VACATION	14	III-II MID-II EXAM (ML&PCC)	12	
THURSDAY	16		16		13		18	SUMMER VACATION	15	III-II MID-II EXAM (ADA&STM)	13	

FRIDAY	17		17		14	AMBEDKAR JAYANTI	19	**SUMMER VACATION**	16	III-II MID-I EXAM(ITE)	14	
SATURDAY	18	**MAHA SHIVARATRI**	18		15		20		17		15	
SUNDAY	19	**HOLIDAY**	19	**HOLIDAY**	16	**HOLIDAY**	21	**HOLIDAY**	18	**HOLIDAY**	16	**HOLIDAY**
MONDAY	20		20		17	III-II(CRT CLASSES)	22	III-II(CRT CLASSES ONLINE)	19		17	BONALU
TUESDAY	21		21		18		23	II-II(CRT CLASSES ONLINE **IV-II MID-II EXAM**	20		18	
WEDNESDAY	22		22	**UGHADI**	19	III-II(CRT CLASSES)	24	III-II(CRT CLASSES ONLINE **IV-II MID-II EXAM**	21		19	GREEN DAY
THURSDAY	23		23		20		25	III-II(CRT CLASSES ONLINE	22		20	
FRIDAY	24		24	**IV-II MID-I EXAM**	21		26	III-II(CRT CLASSES ONLINE) **DEAN,HOD,FACULTY MEETING REG:DEPARMENT WORKS) IV-II PROJECT REVIEW-3**	23		21	Holiday for staff and students due to heavy rain
SATURDAY	25		25	**IV-II MID-I EXAM**	22	RAMZAN	27	II-II(CRT CLASSES ONLINE **(EPSIBA MAM ,FACULTY MEETING REG:ATTAINMENTS)**	24		22	Holiday for staff and students due to heavy rain
SUNDAY	26	**HOLIDAY**	26	**HOLIDAY**	23	**HOLIDAY**	28	**HOLIDAY**	25	**HOLIDAY**	23	**HOLIDAY**
MONDAY	27		27	**IV-II MID-I EXAM**	24		29	III-II(CRT CLASSES)	26	**III-II END EXAM**	24	
TUESDAY	28		28		25		30	III-II(CRT CLASSES)	27		25	
WEDNESDAY			29		26		31	III-II(CRT CLASSES)	28	**III-II END EXAM**	26	
THURSDAY			30	SRIRAMANAVAMI	27				29	**BAKRIDH**	27	Holiday for staff and students due to heavy rain
FRIDAY			31	III-II MID-I EXAM(ML&PC	28				30	**III-II END EXAM**	28	
SATURDAY					29						29	**MUHARRAM**
SUNDAY					30	**HOLIDAY**					30	**HOLIDAY**
MONDAY											31	

CALENDAR INCHARGE **HOD/IT & CSIT** **PRINCIPAL**

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE OUTCOMES (CO'S)

COURSE NAME: COMPILER CONSTRUCTION LAB

Course Name	Course outcomes
C32L1.1	Examine the role of lexical analyzer on the given input data
C32L1.2	Construct Recursive Descent Parser for the given grammar
C32L1.3	Experiment the functionality of non-recursive descent parser(LL(1) by parsing the given input string
C32L1.4	Build the intermediate code from the given source code by using various intermediatecode generation techniques
C32L1.5	Generate the machine code from the given abstract syntax tree of the source code
C32L1.6	Justify the functionality of lexical analyser using LEX, FLEX or JFLEX tool

COURSE ARTICULATION MATRIX

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C32L1.1	3	3	3	-	-	-	-	-	-	-	-	-	2	2	-
C32L1.2	3	2	2	-	-	-	-	-	-	-	-	-	2	2	-
C32L1.3	3	3	2	2	-	-	-	-	-	-	-	-	2	2	-
C32L1.4	3	3	2	2	-	-	-	-	-	-	-	-	2	2	-
C32L1.5	3	3	2	2	-	-	-	-	-	-	-	-	2	2	-
C32L1.6	3	3	2	-	2	-	-	-	-	-	-	-	2	2	-
C22L2	**3**	**2.8**	**2.1**	**2**	**0.6**	-	-	-	-	-	-	-	**2**	**2**	**-**

SRI INDU COLLEGE OF ENGINEERING & TECHNOLOGY
(An Autonomous Institution under UGC, New Delhi)

B.Tech. - III Year – II Semester	L	T	P	C
	0	0	3	1.5

(R20INF32L1) Compiler Construction Lab

Objectives:
To provide an understanding of the language translation peculiarities by designing a complete translator for a mini language.

Recommended System / Software Requirements:

- Intel based desktop PC with minimum of 166 MHZ or faster processor with at least 64MBRAM and 100 MB free disk space
- C++ compiler and JDK kit

EXPERIMENTS

1. This program is to find out whether a given string is an identifier or not.
2. Write a program to simulate a machine known as the Deterministic Finite Automata (DFA).
3. To write a program for dividing the given input program into lexemes.
4. How do you write a program to check the string of a given grammar?
5. Write a program to remove left-recursion from a context-free grammar.
6. Write a program on recursive descent parsing.

Input a grammar from the user. Identify the input Terminals, and Non-terminal

a) Write a C++ Code to compute the FIRST, FOLLOW of all terminal.

b) Write a C++ Code to Compute the LL(1) Parsing Table for the given grammar.

7 . Write a program to show the implementation of Shift-Reduce Parser.

8. Write a program to generate the intermediate code in the form of Polish notation.

9. Write a program for generating for various intermediate code forms:

a) Three address code b) Quadruple.

Course Outcomes:

At the end of the course, the student will be able to

- Examine the role of lexical analyzer on the given input data
- Construct Recursive Descent Parser for the given grammar
- Experiment the functionality of non-recursive descent parser(LL(1) by parsing the given input string
- Build the intermediate code from the given source code by using various intermediatecode generation techniques
- Generate the machine code from the given abstract syntax tree of the source code
- Justify the functionality of lexical analyser using LEX, FLEX or JFLEX tool

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), thosewho come after 5 minutes will not be allowed into the lab.

2. Plan your task properly much before to the commencement, come prepared to the lab with thesynopsis / program / experiment details.

3. Student should enter into the laboratory with:

a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure,Program, Expected Output, etc.,) filled in for the lab session.

b. Laboratory Record updated up to the last session experiments and other utensils (if any) needed inthe lab.

c. Proper Dress code and Identity card.

4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allottedto you by the faculty.

5. Execute your task in the laboratory, and record the results / output in the lab observation note book,and get certified by the concerned faculty.

6. All the students should be polite and cooperative with the laboratory staff, must maintain thediscipline and decency in the laboratory.

7. Computer labs are established with sophisticated and high end branded systems, which should beutilized properly.

8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severepunishment.

9. Students must take the permission of the faculty in case of any urgency to go out ; if anybody foundloitering outside the lab / class without permission during working hours will be treated seriously andpunished appropriately.

10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

Head of the Department **Principal**

Sri Indu College of Engineering &Technology
(An Autonomous Institution under UGC)

Sheriguda (V), Ibrahimpatnam (M), Ranga Reddy (Dist) – 501 510

DEPARTMENT OF INFORMATION TECHNOLOGY

LAB Time - Table

Main Block/First Floor **Lab-1A &1B-CC LAB**

Class: III-II SEM **w.e.f: 02/2/2023**

Time	9:40-10:40	10:40-11:40	11:40-12:40	12:40 To 1:20	1:20-2:15	2:15-3:10	3:10-4:00
Days	1	2	3		4	5	6
Monday				L U N C H			
Tuesday							
Wednesday					CC LAB(III-IT)		
Thursday							
Friday							
Saturday							

HOD DEAN PRINCIPAL

List of Experiments

S.No	Name Of The Experiment	No. of Class required	CO
1.	This program is to find out whether a given string is an identifier or not.	3	CO1
2.	Write a program to simulate a machine known as the Deterministic Finite Automata (DFA).	3	CO1
3.	To write a program for dividing the given input program into lexemes.	3	CO1
4.	How do you write a program to check the string of a given grammar?	3	CO2
5.	Write a program to remove left-recursion from a context-free grammar.	3	C02
6.	Write a program on recursive descent parsing.	3	C02
6(a).	Write a C++ Code to compute the FIRST, FOLLOW of all terminal.	3	C02
6(b).	Write a C++ Code to Compute the LL(1) Parsing Table for the given grammar.	3	C02
7.	Write a program to show the implementation of Shift-Reduce Parser.	3	C02
8.	Write a program to generate the intermediate code in the form of Polish notation.	3	C04
9(a).	Write a program for generating for various intermediate code forms. Three address code	3	C04
9(b).	Write a program for generating for various intermediate code forms. Quadruple.	3	C04
2.ADDITIONAL PROGRAMS			
1.	Write a C program to recognize strings under 'a*', 'a*b+', 'abb'.	3	CO1
2.	Write a C program to simulate lexical analyzer for validating operators.	3	CO1
3.	Write a C program to implement operator precedence parsing.	3	CO2

LAB INCHARGE **HOD**

1. This program is to find out whether a given string is an identifier or not.

RESOURCE:

Turbo C++

PROCEDURE:

Go to debug->run or pressCTRL+F9 to run the program

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
Void main()
{
Char  a[100],str[20],str1[10]={"printf"};
int i,l,flag=0,s;
clrscr();
printf("\n enter the string:");gets(a);
if(strcmp(a,str1)==0)
{
printf("\nnotallow");
}
else
{
l=strlen(a);
if(a[0]=="||a[0]=='@')
{
printf("\ninvalididentifier");
goto p;
}
else
{s=0;
while(s!=l)
{
if(a[s]=="||a[s]=='1'||a[s]=='2'||a[s]=='3'||a[s]=='4'||a[s]=='5'||a[s]=='6'
||a[s]=='7'||a[s]=='8'||a[s]=='9'||a[s]=='0')
{
printf("\ninvalid identifier");
goto p;
}
else
{
flag=0;
}s++;
}
if(flag==1)
{
```

```
printf("\ninvalid identifier");
}
if(flag==0)
{
printf("\nvalid identifier");
}
}
}
p:getch();
}
```

Output:
Enter the string : ab
Valid identifier
Enter the string:12
In valid identifier

Viva Questions

1.What is compiler?
2.What is token?
3.What is difference between token and lexeme?
4.Define phase and pass?
5.What is difference between compiler and interpreter?

2.Write a program to simulate a machine known as the Deterministic Finite Automata(DFA).

RESOURCE:
TurboC++

PROCEDURE:
Gotodebug->runorpressCTRL+F9toruntheprogram

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
struct node{
int  id_num;
int  st_val;
struct node *link0;
struct node *link1;
};
struct node *start, *q, *ptr;
int vst_arr[100], a[10];
int main(){
int count, i, posi, j;
char n[10];
printf("=-=-=-=-=-=-=-=-=-==-=-=-=-=-=-=-=-=-=-=\n");
printf("Enter the number of states in the m/c:");
scanf("%d",&count);
q=(struct node *)malloc(sizeof(struct node)*count);
for(i=0;i<count;i++){
(q+i)->id_num=i;
printf("State Machine::%d\n",i);
printf("Next State if i/p is 0:");
scanf("%d",&posi);
(q+i)->link0=(q+posi);
printf("Next State if i/p is 1:");
scanf("%d",&posi);
(q+i)->link1=(q+posi);
printf("Is the state final state(0/1)?");
scanf("%d",&(q+i)->st_val);
}
printf("Enter the Initial State of the m/c:");
scanf("%d",&posi);
start=q+posi;
printf("=-=-=-=-=-=-=-=-=-==-=-=-=-=-=-=-=-=-=-=\n");
while(1){
printf("=-=-=-=-=-=-=-=-=-==-=-=-=-=-=-=-=-=-=-=\n");
printf("Perform String Check(0/1):");
scanf("%d",&j);
if(j){
```

```
ptr=start;
printf("Enter the string of inputs:");
scanf("%s",n);
posi=0;
while(n[posi]!='\0'){
a[posi]=(n[posi]-'0');
//printf("%c\n",n[posi]);
//printf("%d",a[posi]);
posi++;
}
i=0;
printf("The visited States of the m/c are:");

do{
vst_arr[i]=ptr->id_num;
if(a[i]==0){
ptr=ptr->link0;
}
else if(a[i]==1){
ptr=ptr->link1;
}
else{
printf("iNCORRECT iNPUT\n");
return;
}
printf("[%d]",vst_arr[i]);
i++;
}while(i<posi);
printf("\n");
printf("Present State:%d\n",ptr->id_num);
printf("String Status:: ");
if(ptr->st_val==1)
printf("String Accepted\n");
else
printf("String Not Accepted\n");
}
else
return 0;
}
printf("=-=-=-=-=-=-=-=-=-==-=-=-=-=-=-=-=-=-=-=\n");
return 0;
}
```

Output:

```
Enter the number of states in the m/c:2
State machine:0
Next state if input is 0:1
```

Next state if input is 1:0
Is the state final state(0/1)?1
State machine:1
Next state if input is 0:0
Next state if input is 1:1
Is the state final state(0/1)?0
Enter the initial state of the m/c:0

Perform string check(0/1):1
Enter the string of inputs:1
The visited states of the m/c are:(0)
Present state:0
String status: string Accepted

Viva Questions

1.What is Finite automata?
2.What is Deterministic Finite Automata?
3.What is lexical analyzer?
4.What is Non Deterministic Finite Automata?
5.What are the tuples used in Finite Automata ?

3.To write a program for dividing the given input program into lexemes.

RESOURCE:
Turbo C++

PROCEDURE:
Gotodebug->runorpressCTRL+F9toruntheprogram

PROGRAM:

```
#include<string.h>
#include<ctype.h>
#include<stdio.h>
void keyword(char str[10])
{
   if(strcmp("for",str)==0||strcmp("while",str)==0||strcmp("do",str)==0||strcmp("int",str
)==0||strcmp("float",str)==0||strcmp("char",str)==0||strcmp("double",str)==0||strcmp("static",str)==0||str
cmp("switch",str)==0||strcmp("case",str)==0)
printf("\n%s is a keyword",str);
else
printf("\n%s is an identifier",str);
}
int main()
{
   FILE *f1,*f2,*f3;
charc,str[10],st1[10];
intnum[100],lineno=0,tokenvalue=0,i=0,j=0,k=0;
printf("\nEnter the c Program: ");/*gets(st1);*/
   f1=fopen("input","w");
while((c=getchar())!=EOF)
putc(c,f1);
fclose(f1);
   f1=fopen("input","r");
   f2=fopen("identifier","w");
   f3=fopen("specialchar","w");
while((c=getc(f1))!=EOF)
   {
if(isdigit(c))
      {
tokenvalue=c-'0';
         c=getc(f1);
while(isdigit(c))
         {
tokenvalue*=10+c-'0';
            c=getc(f1);
         }
num[i++]=tokenvalue;
ungetc(c,f1);
```

```
    }
else if(isalpha(c))
      {
putc(c,f2);
         c=getc(f1);
while(isdigit(c)||isalpha(c)||c=='_'||c=='$')
         {
putc(c,f2);
            c=getc(f1);
         }
putc(' ',f2);
ungetc(c,f1);
      }
else if(c==' '||c=='\t')
printf(" ");
else if(c=='\n')
lineno++;
else
putc(c,f3);
   }
fclose(f2);
fclose(f3);
fclose(f1);
printf("\nThe no's in the program are");
for(j=0; j<i; j++)
printf("%d",num[j]);
printf("\n");
   f2=fopen("identifier","r");
   k=0;
printf("The keywords and identifiersare:");
while((c=getc(f2))!=EOF)
   {
if(c!=' ')
str[k++]=c;
else
      {
str[k]='\0';
keyword(str);
         k=0;
      }
   }
fclose(f2);
   f3=fopen("specialchar","r");
printf("\nSpecial characters are");
while((c=getc(f3))!=EOF)
printf("%c",c);
printf("\n");
fclose(f3);
```

```
printf("Total no. of lines are:%d",lineno);
return 0;
}
```

output:

```
Enter the c Program: a+b*c
^Z

The no's in the program are
The keywords and identifiersare:
a is an identifier
b is an identifier
c is an identifier
Special characters are+*
Total no. of lines are:1
```

Viva Questions

1.List the phases of Compiler?

2.What is Grammar?

3.What is symbol table?

4.What are the functions of a Scanner?

5.List various types of Compilers.

4. How do you write a program to check the string of a given grammar?

RESOURCE:

Turbo C++

PROCEDURE:

Go to debug->run or pressCTRL+F9 to run the program

PROGRAM

```
#include<stdio.h>
 #include<conio.h>
 #include<string.h>
 void main()
 char string[50];
 int flag,count=o;
 clrscr();
 printf("The grammar is: S->aS, S->Sb, S->ab\n");
printf("Enter the string to be checked:\n");
gets(string);
 if(string[0]=='a')
 flag=0;
for (count=1;string[count-1]!='\0';count++)
{
if(string[count]=='b')
{
flag=1;
continue;
 }
else if((flag==1)&&(string[count]=='a'))
 {
printf("The string does not belong to the specified grammar");
 break;
}
 else if(string[count]=='a')
continue;
else if(flag==1)&&(string[count]='\0'))
 {
 printf("String accepted…..!!!!");
 break;
 }
 else
{
printf("String not accepted");
 } } }
 getch();
 }
```

output:
The grammar is:
S->aS
S->Sb
S->ab

Enter the string to be checked: aab
String accepted….!!!!

Viva Questions.

1.What is Grammar?

2.What is a parser ?

3.What is YACC?

4.What is the application of Compilers?

5.What are the different types of Parsers?

5.Write a program to remove left recursion from a context free grammar.

RESOURCE:

Turbo C++

PROCEDURE:

Go to debug->run or pressCTRL+F9 to run the program

PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define SIZE 20
int main()
{
   char pro[SIZE], alpha[SIZE], beta[SIZE];
   int nont_terminal,i,j, index=3;
   printf("Enter the Production as E->E|A: ");
   scanf("%s", pro);
   nont_terminal=pro[0];
   if(nont_terminal==pro[index]) //Checking if the Grammar is LEFT RECURSIVE
   {
      //Getting Alpha
      for(i=++index,j=0;pro[i]!='|';i++,j++){
         alpha[j]=pro[i];
         //Checking if there is NO Vertical Bar (|)
         if(pro[i+1]==0){
            printf("This Grammar CAN'T BE REDUCED.\n");
            exit(0); //Exit the Program
         }
      }
      alpha[j]='\0'; //String Ending NULL Character
      if(pro[++i]!=0) //Checking if there is Character after Vertical Bar (|)
      {
         //Getting Beta
         for(j=i,i=0;pro[j]!='\0';i++,j++){
            beta[i]=pro[j];
         }
         beta[i]='\0'; //String Ending NULL character
         //Showing Output without LEFT RECURSION
         printf("\nGrammar Without Left Recursion: \n\n");
         printf(" %c->%s%c'\n", nont_terminal,beta,nont_terminal);
         printf(" %c'->%s%c'|#\n", nont_terminal,alpha,nont_terminal);
      }
      else
         printf("This Grammar CAN'T be REDUCED.\n");
```

```
    }
    else
        printf("\n This Grammar is not LEFT RECURSIVE.\n");
}
```

OUTPUT:

```
Enter the production as E->E|A:
E->E+T|T
Grammar without left recursion:
E->TE'
E'->+TE'|#
```

Viva Questions

1.What is Left Recursion?

2. What is difference between top down and bottom up parsing?

3.What is Abstract syntax tree?

6.Write a program on recursive descent parsing.

RESOURCE:

Turbo C++

PROCEDURE:

Go to debug->run or pressCTRL+F9 to run the program

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<string.>
char input[100];
int i,l;
void main()
{
clrscr();
printf("\nRecursive descent parsing for the following grammar\n");
printf("\nE->TE'\nE'->+TE'/@\nT->FT'\nT'->*FT'/@\nF->(E)/ID\n");
 printf("\nEnter the string to be checked:");
gets(input);
if(E( ))

{

if(input[i+1]=='\0')

printf("\nString is accepted");

else

printf("\nString is not accepted");

}

else

printf("\nString not accepted");

getch( );

}

E( )

{
```

```
if(T( ))
{
if(EP( ))
return(1);
else
return(0);
}
else
return(0);
}
EP( )
{
if(input[i]=='+')
{
i++;
if(T())
{
if(EP())
return(1);
else
return(0);
}
else
return(0);
```

```
}
else
return(1);
}
T()
{
if(F())
{
if(TP())
return(1);
else
return(0);
}
else
return(0);
}
TP()
{
if(input[i]=='*')
{
i++;
if(F( ))
{
if(TP( ))
```

```
return(1);
else
return(0);
}
else
return(0);
}
else
return(1);
}
F()
{
if(input[i]=='(')
{
i++;
if(E())
{
if(input[i]==')')
{
i++;
return(1);
}
else
return(0);
```

```
}
else
return(0);
}
else if(input[i]>='a'&&input[i]<='z'||input[i]>='A'&&input[i]<='Z')
{
i++;
return(1);
}
else
return(0);
}
```

INPUT & OUTPUT:

```
Recursive descent parsing for the following grammar
E->TE'
E'->+TE'/@
T->FT'
T'->*FT'/@
F->(E)/ID
Enter the string to be checked:(a+b)*c
String is accepted
Recursive descent parsing for the following grammar
E->TE'
```

E'->+TE'/@

T->FT'

T'->*FT'/@

F->(E)/ID

Viva Questions.

1.What is recursive descent parsing?

2.What is the meaning of each letter in LR(0)?

3.Define ambiguous grammar?

Input a grammar from the user. Identify the input Terminals, and Non-terminal

a) Write a C++ Code to compute the FIRST, FOLLOW of all terminal.

RESOURCE:

Turbo C++

PROCEDURE:

Go to debug->run or pressCTRL+F9 to run the program

PROGRAM

```
#include<conio.h>
#include<stdio.h>
#include<math.h>
char g[5][2][10]={
 {"e->tE"}, //treat E as e'
 {{"E->+tE"},{"E->N"}}, //treat N as null
 {"t->fT"}, //treat T as t'
 {{"T->*fT"},{"T->N"}},
 {{"f->(e)"},{"f->i"}},
 };
char resf[5][2];
char resfollow[5][5];
int track[5];
void first(char);
void follow(void);
void main()
{
 int i,k;
 clrscr();
 //find first
 for(k=0;k<5;k++)
 {
 for(i=0;i<5;i++)
 track[i]=0;
if(resf[k][0]=='\x0')
 first(g[k][0][0]);
 }
 follow();
 printf("\tFirst\tFollow\n");
 for(i=0;i<5;i++)
 {
 printf("\n\n%c\t",g[i][0][0]);
 printf("%c,%c %10s",resf[i][0],resf[i][1],resfollow[i]);
 }
 getch();
}
```

```
void first(char ref)
{
 int i,j,production,k;
 char c;
 for(i=0;i<5;i++)
 {
 if(g[i][0][0]==ref)
 {
 production=i;
 track[production]=1;
 }
 }
 for(i=0;i<2;i++)
 {
 c=g[production][i][0+3];
 if(c=='+'||c=='*'||c=='('||c==')'||c=='i'||c=='N')
 {
 for(k=0;k<5;k++)
{
 if(track[k]!=0)
 resf[k][i]=c;
 }
 }
 else if(c !='\x0')
 first(c);
 }
}
void follow(void)
{
 int i,j,k,l,pbeta,pB,pA;
 int index[5]={0,0,0,0,0};
 int iA,iB,ibeta;
 int rule3;
 int length,redundant;
 char start,a,c,B,beta,A;
 //rule 1
 for(i=0;i<5;i++)
 {
 for(j=0;j<2;j++)
 {
 start=g[i][j][3];
 if(start != '+' && start != '*' && start != '(' && start != ')' && start != 'N'
&& start != 'i' && start!='\x0')
 resfollow[i][index[i]++]='$';
 }
 //defination
 length=strlen(g[i][0]);
 a=g[i][0][length-1];
```

```
A=g[i][0][length-2];
 if((a=='+'||a=='*'||a=='('||a==')'||a=='i')&&a!='N')
 {
 //search for the index value of A
 for(k=0;k<5;k++)
 if(g[k][0][0]==A)
 pA=k;
 resfollow[pA][index[pA]++]=a;
 }
 }
 //rule 2
 for(j=0;j<5;j++)
 {
 A=g[j][0][0];
 length=strlen(g[j][0]);
 B=g[j][0][length-2];
 beta=g[j][0][length-1];
 for(i=0;i<5;i++)
 {
 if(g[i][0][0]==beta)
 pbeta=i;
 if(g[i][0][0]==A)
 pA=i;
 if(g[i][0][0]==B)
 pB=i;
 }
 if(!(beta=='+' || beta=='*' || beta=='(' || beta==')' || beta=='i') && beta !='N')
 {
 for(i=0;i<2;i++)
 {
 if((resf[pbeta][i])!='N')
{
 //check for redundant element in follow
 redundant=0;
 for(k=0;k<5;k++)
 {
 if(resfollow[pB][k]== resf[pbeta][i])
 redundant=1;
 }
 if(redundant!=1)
 (resfollow[pB][index[pB]++])=(resf[pbeta][i]);
 }
 }
 }
 }
 //rule 3 A->(alpha)(B)(beta)
 for(j=0;j<5;j++)
 {
```

```
A=g[j][0][0];
length=strlen(g[j][0]);
B=g[j][0][length-2];
beta=g[j][0][length-1];
if(!(beta=='+'||beta=='*'||beta=='('||beta==')'||beta=='i'||beta=='N'))
{
for(i=0;i<5;i++)
{
if(g[i][0][0]==beta)
pbeta=i;
if(g[i][0][0]==A)
pA=i;
if(g[i][0][0]==B)
pB=i;
}
for(i=0;i<2;i++)
{
if(resf[pbeta][i]=='N')
{
for(k=0;k<5;k++)
{
//check for redundant element in follow
redundant=0;
for(l=0;l<5;l++)
{
if(resfollow[pB][l]== resfollow[pA][k])
redundant=1;
}
if(redundant!=1)

(resfollow[pB][index[pB]++])=(resfollow[pA][k]);
}
}
}
}
}
//rule 3 A->(alpha)(B)
for(j=0;j<5;j++)
{
A=g[j][0][0];
length=strlen(g[j][0]);
B=g[j][0][length-1];
if(A!=B)
{
for(i=0;i<5;i++)
{
if(g[i][0][0]==A)
pA=i;
```

```
 if(g[i][0][0]==B)
 pB=i;
 }
 if(!(B=='+'||B=='*'||B=='('||B==')'||B=='i'||B=='N'))
 for(i=0;i<5;i++)
 resfollow[pB][index[pB]++]=resfollow[pA][i];
 }
 }
}
```

OUTPUT:

```
 First Follow
e (,i      $,)
E +,N   $,)
t (,i      $,+,)
T *,N   $,+,)
f (,i      $,+,*,)
```

Viva Questions.

1.What is FIRST of a grammar?

2.What is Follow of a grammar?

b)Write a C++ Code to Compute the LL(1) Parsing Table for the given grammar.

RESOURCE:

Turbo C++

PROCEDURE:

Go to debug->run or pressCTRL+F9 to run the program

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
char s[20],stack[20];
 void main()
{
char m[5][6][3]={"tb"," "," ","tb"," "," "," ","+tb"," "," ","n","n","fc"," "," ","fc"," "," "," ","n","*fc","
a","n","n","i"," "," ","(e)"," "," "};
int size[5][6]={2,0,0,2,0,0,0,3,0,0,1,1,2,0,0,2,0,0,0,1,3,0,1,1,1,0,0,3,0,0};
int i,j,k,n,str1,str2;
clrscr();
printf("\n Enter the input string: ");
scanf("%s",s);
strcat(s,"$");
n=strlen(s);
stack[0]='$';
stack[1]='e';
i=1;
j=0;
printf("\nStack Input\n”);
printf(" \n");
while((stack[i]!='$')&&(s[j]!='$'))
{
if(stack[i]==s[j])
{
i--;
j++;
}
switch(stack[i])
{
case 'e': str1=0;
break;
case 'b': str1=1;
break;
case 't': str1=2;
break;
case 'c': str1=3;
```

```
break;
case 'f': str1=4;
break;
}
switch(s[j])
{

case 'i': str2=0;
break;
case '+': str2=1;
break;
case '*': str2=2;
break;
case '(': str2=3;
break;
case ')': str2=4;
break;
case '$': str2=5;
break;
}
if(m[str1][str2][0]=='\0')
{
printf("\nERROR");
exit(0);
}
else if(m[str1][str2][0]=='n')
i--;
else if(m[str1][str2][0]=='i')
stack[i]='i';
else
{
for(k=size[str1][str2]-1;k>=0;k--)
{
stack[i]=m[str1][str2][k];
i++;
}
i--;
}
for(k=0;k<=i;k++)
printf(" %c",stack[k]);
printf(" ");
for(k=j;k<=n;k++)
printf("%c",s[k]);
printf(" \n ");
}
printf("\n SUCCESS");
getch();
}
```

Output:

```
Enter the input string:i*i+i
Stack INPUT
$bt       i*i+i$
$bcf      i*i+i$
$bci      i*i+i$
$bc       *i+i$
$bcf*     *i+i$
$bcf      i+i$
$bci      i+i$
$bc       +i$
$b        +i$
$bt+      +i$
$bt       i$
$bcf      i$
$ bci     i$
$bc       $
$b        $
$         $
```

Viva Questions.

1.What is LL(1) Grammar?

2.What is difference between LR(0) item and LR(1)item?

3.What is difference between LR(0) and SLR(1)?

7.Write a program to show the implementation of Shift-Reduce Parser.

RESOURCE:

Turbo C++

PROCEDURE:

Go to debug->run or pressCTRL+F9 to run the program

PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<string.h>
char ip_sym[15],stack[15];
int ip_ptr=0,st_ptr=0,len,i;
char temp[2],temp2[2];
char act[15];
void check();
void main()
{
clrscr();
printf("\n\t\t SHIFT REDUCE PARSER\n");
printf("\n GRAMMER\n");
printf("\n E->E+E\n E->E/E");
printf("\n E->E*E\n E->a/b");
printf("\n enter the input symbol:\t");
gets(ip_sym);
printf("\n\t stack implementation table");
printf("\n stack \t\t input symbol\t\t action");
printf("\n___\t\t____\t\t_____\n");
printf("\n $\t\t%s$\t\t\t--",ip_sym);
strcpy(act,"shift");
temp[0]=ip_sym[ip_ptr];
temp[1]='\0';
strcat(act,temp);
len=strlen(ip_sym);

for(i=0;i<=len-1;i++)
{
stack[st_ptr]=ip_sym[ip_ptr];
stack[st_ptr+1]='\0';
ip_sym[ip_ptr]=' ';
ip_ptr++;
printf("\n $%s\t\t%s$\t\t\t%s",stack,ip_sym,act);
strcpy(act,"shift");
temp[0]=ip_sym[ip_ptr];
temp[1]='\0';
```

```
strcat(act,temp);
check();
st_ptr++;
}
st_ptr++;
check();
}
void check()
{
int flag=0;
temp2[0]=stack[st_ptr];
temp2[1]='\0';
if((!strcmpi(temp2,"a"))||(!strcmpi(temp2,"b")))
{
stack[st_ptr]='E';

if(!strcmpi(temp2,"a"))
printf("\n $%s\t\t%s$\t\t\tE->a",stack,ip_sym);
else
printf("\n $%s\t\t%s$\t\t\tE->b",stack,ip_sym);
flag=1;
}
if((!strcmpi(temp2,"+"))||(strcmpi(temp2,"*"))||(!strcmpi(temp2,"/")))
{
flag=1;
}
if((!strcmpi(stack,"E+E"))||(!strcmpi(stack,"E\E"))||(!strcmpi(stack,"E*E")))
{
strcpy(stack,"E");
st_ptr=0;
if(!strcmpi(stack,"E+E"))
printf("\n $%s\t\t%s$\t\t\tE->E+E",stack,ip_sym);
else
if(!strcmpi(stack,"E\E"))
printf("\n $%s\t\t%s$\t\t\tE->E\E",stack,ip_sym);
else
if(!strcmpi(stack,"E*E"))
printf("\n $%s\t\t%s$\t\t\tE->E*E",stack,ip_sym);
else
printf("\n $%s\t\t%s$\t\t\tE->E+E",stack,ip_sym);
flag=1;
}

if(!strcmpi(stack,"E")&&ip_ptr==len)
{
printf("\n $%s\t\t%s$\t\t\tACCEPT",stack,ip_sym);
getch();
exit(0);
```

```
}
if(flag==0)
{
printf("\n%s\t\t\t%s\t\t reject",stack,ip_sym);
exit(0);
}
return;
}
```

OUTPUT:

```
SHIFT REDUCE PARSER
GRAMMER
E->E+E
E->E/E
E->E*E
E->a/b
Enter the input symbol: a+b
Stack Implementation Table
Stack Input Symbol Action

------- ----------------- ---------
$       a+b$    --
$a       +b$    shift a
$E       +b$     E->a
$E+      b$     shift +
$E+b     $      shift b
$E+E     $       E->b
$E       $       E->E+E
$E       $       ACCEPT
```

Viva Questions.

1.What is Shift reduce parser?

2.List the different types of bottom up parsing?

3.What are the different error recovery in parsing?

8.Write a program to generate the intermediate code in the form of Polish notation.

RESOURCE:

Turbo C++

PROCEDURE:

Go to debug->run or pressCTRL+F9 to run the program

PROGRAM

```
#include<stdio.h>
#include<ctype.h>
char stack[100];
int top = -1;
void push(char x)
{
stack[++top] = x;
}

char pop()
{
if(top == -1)
return -1;
else

return stack[top--];
}

int priority(char x)
{
if(x == '(')
return 0;
if(x == '+' || x == '-')
return 1;
if(x == '*' || x == '/')
return 2;
return 0;
}

int main()
{
char exp[100];
char *e, x;
printf("Enter the expression : ");
scanf("%s",exp);
printf("\n");
e = exp;
```

```
while(*e != '\0')
{
if(isalnum(*e))
printf("%c ",*e);
else if(*e == '(')
push(*e);
else if(*e == ')')
{

while((x = pop()) != '(')
printf("%c ", x);
}
else
{
while(priority(stack[top]) >= priority(*e))
printf("%c ",pop());
push(*e);
}
e++;
}

while(top != -1)
{
printf("%c ",pop());
}return 0;
}
```

Output:

```
Enter the expression: a+b*c
a b c * +
```

Viva Questions.

1.What is Intermediate code forms.
2.Define SDD?
3.What is difference between syntax and semantics?
4.Define Synthesized attributes?
5.Define Inherited attributes?

9.Write a program for generating for various intermediate code forms:

a)Three address code

RESOURCE:

Turbo C++

PROCEDURE:

Go to debug->run or pressCTRL+F9 to run the program

PROGRAM

```
#include<stdio.h>
#include<string.h>
void pm();
void plus();
void div();
int i,ch,j,l,addr=100;
char ex[10], exp[10] ,exp1[10],exp2[10],id1[5],op[5],id2[5];

void main()
{
clrscr();
while(1)
{
printf("\n1.assignment\n2.arithmetic\n3.relational\n4.Exit\nEnter the choice:");
scanf("%d",&ch);
switch(ch)
{
case 1:
printf("\nEnter the expression with assignment operator:");
scanf("%s",exp);
l=strlen(exp);
exp2[0]='\0';
i=0;
while(exp[i]!='=')
{
i++;
}
strncat(exp2,exp,i);
strrev(exp);
exp1[0]='\0';
strncat(exp1,exp,l-(i+1));
strrev(exp1);
printf("Three address code:\ntemp=%s\n%s=temp\n",exp1,exp2);
break;
```

```
case 2:
printf("\nEnter the expression with arithmetic operator:");

scanf("%s",ex);
strcpy(exp,ex);
l=strlen(exp);
exp1[0]='\0';

for(i=0;i<l;i++)
{
if(exp[i]=='+'||exp[i]=='-')
{
if(exp[i+2]=='/'||exp[i+2]=='*')
{
pm();
break;
}
else
{
plus();
break;
}
}
else if(exp[i]=='/'||exp[i]=='*')
{
div();
break;
}
}
break;

case 3:

printf("Enter the expression with relational operator");
scanf("%s%s%s",&id1,&op,&id2);
if(((strcmp(op,"<")==0)||(strcmp(op,">")==0)||(strcmp(op,"<=")==0)||(strcmp(op,">=")==0)||(strcm
p(op,"==")==0)||(strcmp(op,"!=")==0))==0)
printf("Expression is error");
else
{
printf("\n%d\tif %s%s%s goto %d",addr,id1,op,id2,addr+3);
addr++;
printf("\n%d\t T:=0",addr);
addr++;
printf("\n%d\t goto %d",addr,addr+2);
```

```
addr++;
printf("\n%d\t T:=1",addr);
}
break;
case 4:
exit(0);
}
}
}
void pm()
{
strrev(exp);
j=l-i-1;
strncat(exp1,exp,j);
strrev(exp1);
printf("Three address code:\ntemp=%s\ntemp1=%c%ctemp\n",exp1,exp[j+1],exp[j]);
}

void div()
{
strncat(exp1,exp,i+2);
printf("Three address code:\ntemp=%s\ntemp1=temp%c%c\n",exp1,exp[i+2],exp[i+3]);
}
void plus()
{
strncat(exp1,exp,i+2);
printf("Three address code:\ntemp=%s\ntemp1=temp%c%c\n",exp1,exp[i+2],exp[i+3]);
}
```

Output:

```
1. assignment
2. arithmetic
3. relational
4. Exit
Enter the choice:1
Enter the expression with assignment operator:
a=b
Three address code:
temp=b
a=temp

1.assignment

2.arithmetic
3.relational
4.Exit
Enter the choice:2
Enter the expression with arithmetic operator:
```

```
a+b-c
Three address code:
temp=a+b
temp1=temp-c

1.assignment
2.arithmetic
3.relational
4.Exit
Enter the choice:2
Enter the expression with arithmetic operator:
a-b/c
Three address code:
temp=b/c
temp1=a-temp

1.assignment
2.arithmetic
3.relational
4.Exit
Enter the choice:2
Enter the expression with arithmetic operator:
a*b-c
Three address code:

temp=a*b
temp1=temp-c

1.assignment
2.arithmetic
3.relational
4.Exit
Enter the choice:2
Enter the expression with arithmetic operator:a/b*c
Three address code:
temp=a/b
temp1=temp*c
1.assignment
2.arithmetic
3.relational
4.Exit
Enter the choice:3
Enter the expression with relational operator
a
<=
b
100 if a<=b goto 103
101 T:=0
```

102 goto 104
103 T:=1
1.assignment
2.arithmetic
3.relational

4.Exit
Enter the choice:4

Viva Questions.
1.What is three address code?
2.What is cross compiler?
3.What are the different types of bottom up parsers?

b) Quadruple

RESOURCE:

Turbo C++

PROCEDURE:

Go to debug->run or pressCTRL+F9 to run the program

PROGRAM

```
#include <bits/stdc++.h>
using namespace std;
/* Function to find a maximum product of a
quadruple in array of integers of size n */
int maxProduct(int arr[], int n)
{
// if size is less than 4, no quadruple exists
if (n < 4)
return -1;
// will contain max product
int max_product = INT_MIN;
for (int i = 0; i < n - 3; i++)
for (int j = i + 1; j < n - 2; j++)
for (int k = j + 1; k < n - 1; k++)
for (int l = k + 1; l < n; l++)
max_product = max(max_product,
arr[i] * arr[j] * arr[k] * arr[l]);
return max_product;
}
// Driver program to test above functions
int main()
{
int arr[] = { 10, 3, 5, 6, 20 };
int n = sizeof(arr) / sizeof(arr[0]);
int max = maxProduct(arr, n);
if (max == -1)
cout << "No Quadruple Exists";
else
cout << "Maximum product is " << max;
return 0;
}
```

Output:
Maximum product is 6000 Viva Questions

Viva questions
1.What is Quadruple?
2.What is Triple?
3.What is LALR?

ADDITIONAL PROGRAMS

1.Write a C program to recognize strings under 'a*', 'a*b+', 'abb'.

RESOURCE:
Turbo C++

PROGRAM LOGIC:
By using transition diagram we verify input of the state.If the state recognize the given pattern rule. Then print string is accepted under a*/ a*b+/ abb.Else print string not accepted.
PROCEDURE:
Go to debug -> run or press CTRL + F9 to run the program.

PROGRAM:

```
#include<stdioh>
#include<conio.h>
 #include<string.h>
#include<stdlib.h>
void main()
{
char s[20],c;
int state=0,i=0;
clrscr();
printf("\n Enter a string:");gets(s);
while(s[i]!='\0')
{
switch(state)

case 0: c=s[i++];
if(c=='a')
state=1;
else if(c=='b')
state=4;
else
state=6;
break; case 2: c=s[i++];
if(c=='a')
state=6;else if(c=='b')
state=2;
else
state=6;
break;
case 3: c=s[i++];
if(c=='a')
state=3;else if(c=='b')
state=2;
else
if(c=='a')
```

```
state=6;
else if(c=='b')
state=5;
else
state=6;
break; case 5: c=s[i++];
if(c=='a')
state=6;
else if(c=='b')
state=2;
else
state=6;break;
case 6: printf("\n %s is not recognised.",s);exit(0);
}
}

f(state==1)
printf("\n %s is accepted unde rule 'a'",s);else if((state==2)||(state==4))
printf("\n %s is accepted under rule 'a*b+'",s);else if(state==5)
printf("\n %s is accepted under rule 'abb'",s);
getch();
}
```

INPUT & OUTPUT:

Input :

```
Enter a String: aaaabbbbb
Output:

aaaabbbbb is accepted under rule 'a*b+'

Enter a string: cdgs cdgs is not recognized
```

2. Write a C program to simulate lexical analyzer for validating operators.

RESOURCE:
Turbo C++

PROGRAM LOGIC :
Read the given input.
If the given input matches with any operator symbol.
Then display in terms of words of the particular symbol.Else print not a operator.

PROCEDURE:
Go to debug -> run or press CTRL + F9 to run the program.

PROGRAM:

```
#include<stdio.h>
 #include<conio.h>
void main()
{
char s[5];
clrscr();
printf("\n Enter any operator:");
gets(s);
switch(s[0])
{
case'>': if(s[1]=='=')
printf("\n Greater than or equal");
else
printf("\n Greater than");
break;
case'<': if(s[1]=='=')
printf("\n Less than or equal");
else
printf("\nLess than");
break;
case'=': if(s[1]=='=')
printf("\nEqual to");
 else
printf("\nAssignment");
break;
case'!':  if(s[1]=='=')
printf("\nNot Equal");
else
printf("\n Bit Not");
break;
case'&': if(s[1]=='&')
printf("\nLogical AND");
else
printf("\n Bitwise AND");
break;
case'|': if(s[1]=='|')
printf("\nLogical OR");
else
printf("\nBitwise OR");
break;
```

```
case'+': printf("\n Addition");
break;
case'-': printf("\nSubstraction")
;break;
case'*': printf("\nMultiplication");
break;
case'/':  printf("\nDivision");
break;
case'%': printf("Modulus");
break;
default: printf("\n Not a operator");
}
getch();
}
```

INPUT & OUTPUT:Input
Enter any operator: *
Output Multiplication

3.Write a C program to implement operator precedence parsing.

RESOURCE:
Turbo C++

PROGRAM LOGIC:

Read the arithmetic input string.
Verify the precedence between terminals and symbols
Find the handle enclosed in < . > and reduce it to production symbol.
Repeat the process till we reach the start node.

PROCEDURE:
Go to debug -> run or press CTRL + F9 to run the program.

PROGRAM:

```
#include<stdio.h>
 char str[50],opstr[75];
int  f[2][9]={2,3,4,4,4,0,6,6,0,1,1,3,3,5,5,0,5,0};
int col,col1,col2;char c;
swt()
{
switch(c)
{
case'+':col=0
;break;case'-':col=1;
break; case'*':col=2;
break;case'/':col=3;
break;
 case'^':col=4;
break;
case'(':col=5;
break; case')':col=6;
break;
case'd':col=7;
break;case'$':col=8;
break;
default:printf("\nTERMINAL MISSMATCH\n");
exit(1);
break;
}
// return 0;
}
main()
{
int i=0,j=0,col1,cn,k=0;int t1=0,foundg=0;
char temp[20];clrscr();
printf("\nEnter arithmetic expression:");scanf("%s",&str);
while(str[i]!='\0')
i++;
```

```
str[i]='$';
str[++i]='\0';
printf("%s\n",str);come:
i=0;
opstr[0]='$';j=1;
c='$';
swt(); col1=col;c=str[i];
swt(); col2=col;
if(f[1][col1]>f[2][col2])
{
opstr[j]='>';j++;
}
else if(f[1][col1]<f[2][col2])
{
opstr[j]='<';j++;
}
else
{

}

opstr[j]='=';j++;

while(str[i]!='$')
{
c=str[i];
swt(); col1=col; c=str[++i];swt(); col2=col;
opstr[j]=str[--i];j++;
if(f[0][col1]>f[1][col2])
{
opstr[j]='>';j++;
}
else if(f[0][col1]<f[1][col2])
{
opstr[j]='<';j++;
}
opstr[j]='$';
}
else
{
opstr[j]='=';j++;
} i++;
opstr[++j]='\0';
printf("\nPrecedence Input:%s\n",opstr);i=0;
j=0;
while(opstr[i]!='\0')
                {
foundg=0; while(foundg!=1)
{
if(opstr[i]=='\0')goto redone;if(opstr[i]=='>')
```

```
foundg=1; t1=i;
i++;
}
if(foundg==1) for(i=t1;i>0;i--)
if(opstr[i]=='<')break; if(i==0){printf("\nERROR\n");exit(1);} cn=i;
j=0; i=t1+1;
while(opstr[i]!='\0')
{
temp[j]=opstr[i]; j++;i++;
}
temp[j]='\0';
opstr[cn]='E'; opstr[++cn]='\0'; strcat(opstr,temp); printf("\n%s",opstr);i=1;
}
redone:k=0; while(opstr[k]!='\0')
{
k++;
if(opstr[k]=='<')
{
Printf("\nError");exit(1);
}
}
if((opstr[0]=='$')&&(opstr[2]=='$'))goto sue;i=1
while(opstr[i]!='\0')
{
c=opstr[i]; if(c=='+'||c=='*'||c=='/'||c=='$')
{
temp[j]=c;j++;}i++;
}
temp[j]='\0'; strcpy(str,temp);goto come;
sue:
printf("\n success");return 0;
}
```

INPUT & OUTPUT:

```
Enter the arithmetic expression(d*d)+d$
Output:
(d*d)+d$
Precedence input:$<(<d>*<d>)>+<d>$
$<(E*<d>)>+<d>$
$<(E*E)>+<E>$
$E+<E>$
$E+E$
Precedence input:$<+>$$E$
success
```

SRI INDU COLLEGE OF ENGINEERING & TECHNOLOGY
(An Autonomous Institution under UGC, New Delhi)
Recognized under 2(f) and 12(B) of UGC Act 1956
NBA Accredited, Approved by AICTE and Permanently affiliated to JNTUH
Sheriguda (V), Ibrahimpatnam, R.R.Dist, Hyderabad - 501 510

2001

D4

BR-20

BTECH III-II SEM –LAB INTERNAL EXAM-1 QUESTION PAPER(2022-2023)

DEPARTMENT OF INFORMATION TECHNOLOGY

(R20INF32L1)COMPILER CONSTRUCTION LAB

BATCH:2020-2024 **DATE:6/4/2023(FN)**

SET-I

1) This program is to find out whether a given string is an identifier or not.
2) How do you write a program to check the string of a given grammar?

SET-II

1) Write a program to simulate a machine known as the Deterministic Finite Automata (DFA).

SET-III

1)To write a program for dividing the given input program into lexemes.

SET-IV

1) This program is to find out whether a given string is an identifier or not.
2) Write a program to remove left-recursion from a context-free grammar.

SRI INDU COLLEGE OF ENGINEERING & TECHNOLOGY
(An Autonomous Institution under UGC, New Delhi)
Recognized under 2(f) and 12(B) of UGC Act 1956
NBA Accredited, Approved by AICTE and Permanently affiliated to JNTUH
Sheriguda (V), Ibrahimpatnam, R.R.Dist, Hyderabad - 501 510

D4

BR-20

BTECH III-II SEM –LAB INTERNAL EXAM-2 QUESTION PAPER(2022-2023)

DEPARTMENT OF INFORMATION TECHNOLOGY

(R20INF32L1)COMPILER CONSTRUCTION LAB

BATCH:2020-2021 **DATE:9/6/2023(AN)**

1.Write a C++ Code to Compute the LL(1) Parsing Table for the given grammar.

2.Write a program to show the implementation of Shift-Reduce Parser.

3. Write a program to generate the intermediate code in the form of Polish notation.

4. Write a program for generating for various intermediate code forms

by using three address code

5. Write a program for generating for various intermediate code forms by using Quadruple

SRI INDU COLLEGE OF ENGINEERING & TECHNOLOGY
(An Autonomous Institution under UGC, New Delhi)
Recognized under 2(f) and 12(B) of UGC Act 1956
NBA Accredited, Approved by AICTE and Permanently affiliated to JNTUH
Sheriguda (V), Ibrahimpatnam, R.R.Dist, Hyderabad - 501 510
2001

D4

BR-20

BTECH III-II SEM –LAB EXTERNAL EXAM QUESTION PAPER(2022-2023)

DEPARTMENT OF INFORMATION TECHNOLOGY

COMPILER CONSTRUCTION LAB (R20INF32L1)

BATCH: 2020-2024 **DATE:17/6/2023**

SET-1

1. This program is to find out whether a given string is an identifier or not.
2. Write a C++ Code to Compute the LL(1) Parsing Table for the given grammar.

SET-2

1. Write a program to simulate a machine known as the Deterministic Finite Automata(DFA)
2. Write a program to show the implementation of Shift-Reduce Parser.

SET-3

1. To write a program for dividing the given input program into lexemes.
2. Write a program to generate the intermediate code in the form of Polish notation.

SET-4

1. How do you write a program to check the string of a given grammar?
2. Write a program for generating for various intermediate code forms by using Three address code

SET-5

1. Write a program to remove left-recursion from a context-free grammar
2. Write a program for generating for various intermediate code forms by using Quadruple

SET-6

1. Write a program on recursive descent parsing.
2. Write a program to show the implementation of Shift-Reduce Parser.