CDSS Lab Programs
1a. Program to count the number of characters, words, spaces and lines in a given input file.
1b. Program to recognize and count the number of identifiers in a file.
2a. Program to count the numbers of comment lines in a given C program.Also eliminate them and copy the resulting program into separate file.
2b. Program to recognize whether a given sentence is simple or compound.
3a. Program to count no of: i.+ve and –ve integers ii. +ve and –ve fractions
3b. Program to count the no of ‘scan f’ and ‘printf’ statements in a C program. Replace them with ‘readf’ and ‘writef’ statements respectively.
4.Program to evaluate arithmetic expression involving operators +,-,*,/
5.Program to recognize a valid variable which starts with a letter, followed by any number of letters or digits.
6. Program to recognize the strings using the grammar ($a^n b^n$;n>=0)
7. C Program to implement Pass1 of Assembler
8. C Program to implement Absolute Loader
9. C program to find the FIRST in context free grammar.
10.C Program to implement Shift Reduce Parser for the given grammar E →E+E E→E*E E→(E) E → id
11.C Program to implement code optimization techniques.

COMPILER DESIGN PROGRAMS:

7. C Program to implement Pass 1 algorithm of assembler.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main(){
FILE *f1,*f2,*f3,*f4;
int lc,sa,l,op1,o,len;
char m1[20],la[20],op[20],otp[20];
clrscr();
f1=fopen("input.txt","r");
f3=fopen("symtab.txt","w");
fscanf(f1,"%s %s %d",la,m1,&op1);
if(strcmp(m1,"START")==0)
{
sa=op1;
lc=sa;
printf("\t%s\t%s\t%d\n",la,m1,op1);
}
else
lc=0;
fscanf(f1,"%s %s",la,m1);
while(!feof(f1))
{
fscanf(f1,"%s",op);
printf("\n%d\t%s\t%s\t%s\n",lc,la,m1,op);
if(strcmp(la,"-")!=0)
{
fprintf(f3,"\n%d\t%s\n",lc,la);
}
f2=fopen("optab.txt","r");
fscanf(f2,"%s %d",otp,&o);
while(!feof(f2))
{
if(strcmp(m1,otp)==0)
{
lc=lc+3;
break;
}
fscanf(f2,"%s %d",otp,&o);
}
fclose(f2);
if(strcmp(m1,"WORD")==0)
{
lc=lc+3;
}
```

```
else if(strcmp(m1,"RESW")==0)
{
op1=atoi(op);
lc=lc+(3*op1);
}
else if(strcmp(m1,"BYTE")==0)
{
if(op[0]=='X')
lc=lc+1;
else
{
len=strlen(op)-2;
lc=lc+len;}
}
else if(strcmp(m1,"RESB")==0)
{
op1=atoi(op);
lc=lc+op1;
}
fscanf(f1,"%s%s",la,m1);
}
if(strcmp(m1,"END")==0)
{
printf("Program length =\n%d",lc-sa);
}
fclose(f1);
fclose(f3);
getch();
}
```

Input.txt		
COPY	START	1000
-	LDA	ALPHA
-	ADD	ONE
-	SUB	TWO
-	STA	BETA
ALPHA	BYTE	C'KLNCE
ONE	RESB	2
TWO	WORD	5
BETA	RESW	1
-	END	-

Optab.txt	
LDA	00
STA	23
ADD	01
SUB	05

Output:

Symtab.txt	
1012	ALPHA
1017	ONE
1019	TWO
1022	BETA

COPY	**START**	**1000**	
1000	-	LDA	ALPHA
1003	-	ADD	ONE
1006	-	SUB	TWO
1009	-	STA	BETA
1012	ALPHA	BYTE	C'KLNCE
1017	ONE	RESB	2
1019	TWO	WORD	5
1022	BETA	RESW	1
1025	-	END	-
Program length = 25			

8. C Program to implement Absolute Loader.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
void main()
{
FILE *fp;
int i,addr1,l,j,staddr1;
char name[10],line[50],name1[10],addr[10],rec[10],ch,staddr[10];
clrscr();
printf("enter program name:" );
scanf("%s",name);
fp=fopen("abssrc.txt","r");
fscanf(fp,"%s",line);
for(i=2,j=0;i<8,j<6;i++,j++)
name1[j]=line[i];
name1[j]='\0';
printf("name from obj. %s\n",name1);
if(strcmp(name,name1)==0)
{
do
{
fscanf(fp,"%s",line);
if(line[0]=='T')
{
for(i=2,j=0;i<8,j<6;i++,j++)
staddr[j]=line[i];
staddr[j]='\0';
staddr1=atoi(staddr);
i=12;
while(line[i]!='$')
{
if(line[i]!='^')
{
printf("00%d \t %c%c\n", staddr1,line[i],line[i+1]);
staddr1++;
i=i+2;
}
else i++;
}
}
else if(line[0]='E')
fclose(fp);
```

```
}while(!feof(fp));
}
getch();
}
```

INPUT (ABSSRC.TXT)
H^SAMPLE^001000^0035
T^001000^0C^001003^071009$
T^002000^03^111111$
E^001000

OUTPUT
enter program name: SAMPLE
name from obj. SAMPLE
001000 00
001001 10
001002 03
001003 07
001004 10
001005 09
002000 11
002001 11
002002 11

9.C program to find the FIRST in context free grammar.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>

void main()
{

        char t[5],int[10],p[5][5],first[5][5],temp;
        int i,j,not,nont,k=0,f=0;
        clrscr();

        printf("\nEnter the no. of Non-terminals in the grammer:");
        scanf("%d",&nont);
        printf("\nEnter the Non-terminals in the grammer:\n");
        for(i=0;i<nont;i++)
        {
                scanf("\n%c",&nt[i]);
        }

        printf("\nEnter the no. of Terminals in the grammer: ( Enter { for absiline )
");
        scanf("%d",&not);
        printf("\nEnter the Terminals in the grammer:\n");
        for(i=0;i<not||t[i]=='$';i++)
        {
                scanf("\n%c",&t[i]);
        }

    for(i=0;i<nont;i++)
    {
            p[i][0]=nt[i];
            first[i][0]=nt[i];
    }

    printf("\nEnter the productions :\n");
    for(i=0;i<nont;i++)
```

```
        {
                scanf("%c",&temp);
                printf("\nEnter the production for %c ( End the production with '$'
sign ) :",p[i][0]);
                for(j=0;p[i][j]!='$';)
                {
                        j+=1;
                        scanf("%c",&p[i][j]);
                }
        }

        for(i=0;i<nont;i++)
        {
                printf("\nThe production for %c -> ",p[i][0]);
                for(j=1;p[i][j]!='$';j++)
                {
                        printf("%c",p[i][j]);
                }
        }
```

```
for(i=0;i<nont;i++)
{
        f=0;
        for(j=1;p[i][j]!='$';j++)
        {
                for(k=0;k<not;k++)
                {
                        if(f==1)
                        break;

                        if(p[i][j]==t[k])
                        {
                                first[i][j]=t[k];
                                first[i][j+1]='$';
                                f=1;
                                break;
                        }

                        else if(p[i][j]==nt[k])
```

```
                        {
                                first[i][j]=first[k][j];
                                if(first[i][j]=='{')
                                continue;
                                first[i][j+1]='$';
                                f=1;
                                break;
                        }
                }
        }
}

for(i=0;i<nont;i++)
{
        printf("\n\nThe first of %c -> ",first[i][0]);
        for(j=1;first[i][j]!='$';j++)
        {
                printf("%c\t",first[i][j]);
        }
}

getch();
}
```

10.C Program to implement Shift Reduce Parser for the given grammar

E →E+E
E→E*E
E→(E)
E → id

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int k=0,z=0,i=0,j=0,c=0;
char a[16],ac[20],stk[15],act[10];
void check();
void main()
   {
      clrscr();
      puts("GRAMMAR is E->E+E \n E->E*E \n E->(E) \n E->id");
      puts("enter input string ");
      gets(a);
      c=strlen(a);
      strcpy(act,"SHIFT->");
      puts("stack \t input \t action");
      for(k=0,i=0; j<c; k++,i++,j++)
       {
         if(a[j]=='i' && a[j+1]=='d')
           {
              stk[i]=a[j];
              stk[i+1]=a[j+1];
              stk[i+2]='\0';
              a[j]=' ';
              a[j+1]=' ';
              printf("\n$%s\t%s$\t%sid",stk,a,act);
              check();
           }
         else
           {
              stk[i]=a[j];
              stk[i+1]='\0';
              a[j]=' ';
              printf("\n$%s\t%s$\t%ssymbols",stk,a,act);
              check();
           }
       }
       getch();
   }
void check()
   {
```

```
  strcpy(ac,"REDUCE TO E");
  for(z=0; z<c; z++)
    if(stk[z]=='i' && stk[z+1]=='d')
      {
        stk[z]='E';
        stk[z+1]='\0';
        printf("\n$%s\t%s$\t%s",stk,a,ac);
        j++;
      }
  for(z=0; z<c; z++)
    if(stk[z]=='E' && stk[z+1]=='+' && stk[z+2]=='E')
      {
        stk[z]='E';
        stk[z+1]='\0';
        stk[z+2]='\0';
        printf("\n$%s\t%s$\t%s",stk,a,ac);
        i=i-2;
      }
  for(z=0; z<c; z++)
    if(stk[z]=='E' && stk[z+1]=='*' && stk[z+2]=='E')
      {
        stk[z]='E';
        stk[z+1]='\0';
        stk[z+1]='\0';
        printf("\n$%s\t%s$\t%s",stk,a,ac);
        i=i-2;
      }
  for(z=0; z<c; z++)
    if(stk[z]=='(' && stk[z+1]=='E' && stk[z+2]==')')
      {
        stk[z]='E';
        stk[z+1]='\0';
        stk[z+1]='\0';
        printf("\n$%s\t%s$\t%s",stk,a,ac);
        i=i-2;
      }
}
```


GRAMMAR is E=E+E
E=E*E
E=(E)
E=id
input string is
(id*id)+id
stack input action
$(id*id)+id$ SHIFT ->symbols
$(id *id)+id$ SHIFT ->id
$(E *id)+id$ REDUCE TO E
$(E* id)+id$ SHIFT ->symbols
$(E*id)+id$ SHIFT ->id
$(E*E)+id$ REDUCE TO E
$(E)+id$ REDUCE TO E
$(E) +id$ SHIFT ->symbols
$E +id$ REDUCE TO E
$E+ id$ SHIFT ->symbols
$E+id $ SHIFT ->id
$E+E $ REDUCE TO E
$E $ REDUCE TO E_