Syllabus

IT2307 SYSTEM SOFTWARE LAB 0 0 3 2

***(*Using C)**

Implement a symbol table with functions to create, insert, modify, search, and display.

1. Implement pass one of a two pass assembler.

2. Implement pass two of a two pass assembler.

Implement a single pass assembler.

Implement a two pass macro processor

Implement a single pass macro processor.

3. Implement an absolute loader.

4. Implement a relocating loader.

5. Implement pass one of a direct-linking loader.

6. Implement pass two of a direct-linking loader.

7. Implement a simple text editor with features like insertion / deletion of a character, word, and sentence.

8. Implement a symbol table with suitable hashing

SYMBOL TABLE

AIM:

To implement a Symbol table with functions to create, insert, modify, search and display in C language.

ALGORITHM:

1. Start the program
2. Define the structure of the symbol table
3. Enter the choice for performing the operations in the symbol table
4. If choice is 1, search symbol table for the symbol to be inserted. If the symbol is already present display "Duplicate Symbol", else insert symbol and corresponding address in the symbol table
5. If choice is 2, symbols present in the symbols table are displayed
6. If choice is 3, symbol to be deleted is searched in the symbol table, if found deletes else displays "Not Found".
7. If choice is 5, the symbol to be modified is searched in the symbol table. The label or address or both can be modified.

PROGRAM:

```
/*              C Program to implement SYMBOL TABLE              */

#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<string.h>
#define null 0
int size=0;
void insert();
void del();
int search(char lab[]);
void modify();
void display();
struct symbtab
{
char label[10];
int addr;
struct symbtab *next;
};
struct symbtab *first,*last;
void main()
{
int op;
int y;
char la[10];
clrscr();
do
```

```
{
printf("\nSYMBOL TABLE IMPLEMENTATION\n");
printf("1.INSERT\n");
printf("2.DISPLAY\n");
printf("3.DELETE\n");
printf("4.SEARCH\n");
printf("5.MODIFY\n");
printf("6.END\n");
printf("\nEnter your option: ");
scanf("%d",&op);
switch(op)
{
case 1:
insert();
display();
break;
case 2:
display();
break;
case 3:
del();
display();
break;
case 4:
printf("Enter the label to be searched: ");
scanf("%s",la);
y=search(la);
if(y==1)
printf("The label is already in the symbol table\n");
else
printf("The label is not found in the symbol tablel\n");
break;
case 5:
modify();
display();
break;
case 6:
break;
}
}while(op<6);
getch();
}
void insert()
{
int n;
char l[10];
printf("Enter the label: ");
scanf("%s",l);
n=search(l);
if(n==1)
printf("The label is already in the symbol table. Duplicate cant be inserted\n");
```

```
else
{
struct symbtab *p;
p=malloc(sizeof(struct symbtab));
strcpy(p->label,l);
printf("Enter the address: ");
scanf("%d",&p->addr);
p->next=null;
if(size==0)
{
first=p;
last=p;
}
else
{
last->next=p;
last=p;
}
size++;
}
}
void display()
{
int i;
struct symbtab *p;
p=first;
printf("LABEL\tADDRESS\n");
for(i=0;i<size;i++)
{
printf("%s\t%d\n",p->label,p->addr);
p=p->next;
}
}
int search(char lab[])
{
int i,flag=0;
struct symbtab *p;
p=first;
for(i=0;i<size;i++)
{
if(strcmp(p->label,lab)==0)
{
flag=1;
}
p=p->next;
}
return flag;
}
void modify()
{
char l[10],nl[10];
```

```
int add,choice,i,s;
struct symbtab *p;
p=first;
printf("What do you want to modify?\n");
printf("1.Only the label\n");
printf("2.Only the address of a particular label\n");
printf("3.Both the label and address\n");
printf("Enter your choice: ");
scanf("%d",&choice);
switch(choice)
{
case 1:
printf("Enter the old label\n");
scanf("%s",l);
printf("Enter the new label: ");
scanf("%s",nl);
s=search(l);
if(s==0)
printf("\nNo such label");
else
{
for(i=0;i<size;i++)
{
if(strcmp(p->label,l)==0)
{
strcpy(p->label,nl);
}
p=p->next;
}
}
break;
case 2:
printf("Enter the label whose address is to be modified: ");
scanf("%s",l);
printf("Enter the new address: ");
scanf("%d",&add);
s=search(l);
if(s==0)
printf("\nNo such label");
else
{
for(i=0;i<size;i++)
{
if(strcmp(p->label,l)==0)
{
p->addr=add;
}
p=p->next;
}
}
break;
```

```
case 3:
printf("Enter the old label: ");
scanf("%s",l);
printf("Enter the new label: ");
scanf("%s",nl);
printf("Enter the new address: ");
scanf("%d",&add);
s=search(l);
if(s==0)
printf("\nNo such label");
else
{
for(i=0;i<size;i++)
{
if(strcmp(p->label,l)==0)
{
strcpy(p->label,nl);
p->addr=add;
}
p=p->next;
}
}
break;
}
}
void del()
{
int a;
char l[10];
struct symbtab *p,*q;
p=first;
printf("Enter the label to be deleted: ");
scanf("%s",l);
a=search(l);
if(a==0)
printf("Label not found\n");
else
{
if(strcmp(first->label,l)==0)
first=first->next;
else if(strcmp(last->label,l)==0)
{
q=p->next;
while(strcmp(q->label,l)!=0)
{
p=p->next;
q=q->next;
}
p->next=null;
last=p;
}
```

```
else
{
q=p->next;
while(strcmp(q->label,l)!=0)
{
p=p->next;
q=q->next;
}
p->next=q->next;
}
size--;
}
}
```

RESULT:

Thus a Symbol table with functions to create, insert, modify, search and display is implemented in C language.

Ex.No: 02

PASS ONE OF TWO PASS ASSEMBLER

AIM:

To implement pass one of a two pass assembler in C language.

ALGORITHM:

1. Open and Read the input file
2. If the input line has the opcode "START" do the following
   - 2.1 Find if there is any operand field after "START", initialize the LOCCTR to the operand value
   - 2.2 Otherwise if there is no value in the operand field then LOCCTR is set to 0
3. Write the input line to the intermediate file
4. Do the following steps until the opcode is END
   - 4.1 Check the Symbol table, if the symbol is not available then enter that symbol into the SYMTAB, along with the memory address in which it is stored.Otherwise, the error message should be displayed
   - 4.2 If there is a opcode
     - 4.2.1 If opcode is present in the OPTAB, then increment the LOCCTR by 3
     - 4.2.2 If opcode is "WORD", then increment LOCCTR by 3;
     - 4.2.3 If opcode is "BYTE", then increment LOCCTR by 1;
     - 4.2.4 If opcode is "RESW" then increment LOCCTR by the integer equivalent of the operand value * 3;
     - 4.2.5 If opcode is "RESB", then increment LOCCTR by the integer equivalent of the operand value

4.3 Write the processed lines in the intermediate file along with their location counters

5. To find the length of the program, Subtract the starting address of the program from the final value of the LOCCTR
6. Close all the files and exit

PROGRAM:

```
/*      C Program to implement PASS ONE OF TWO PASS ASSEMBLER      */
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
void main()
{
char opcode[10],mnemonic[10],operand[10],label[10],code[10];
int locctr,start,length;
FILE *fp1,*fp2,*fp3,*fp4;
clrscr();
fp1=fopen("input.dat","r");
fp2=fopen("symtab.dat","w");
fp3=fopen("out.dat","w");
```

```
fp4=fopen("optab.dat","r");
fscanf(fp1,"%s%s%s",label,opcode,operand);
if(strcmp(opcode,"START")==0)
{
start=atoi(operand);
locctr=start;
fprintf(fp3,"%s\t%s\t%s\n",label,opcode,operand);
fscanf(fp1,"%s%s%s",label,opcode,operand);
}
else
locctr=0;
while(strcmp(opcode,"END")!=0)
{
fprintf(fp3,"%d\t",locctr);
if(strcmp(label,"**")!=0)
fprintf(fp2,"%s\t%d\n",label,locctr);
rewind(fp4);
fscanf(fp4,"%s",mnemonic);
while(strcmp(mnemonic,"END")!=0)
{
if(strcmp(opcode,mnemonic)==0)
{
locctr+=3;
break;
}
fscanf(fp4,"%s",mnemonic);
}
if(strcmp(opcode,"WORD")==0)
locctr+=3;
else if(strcmp(opcode,"RESW")==0)
locctr+=(3*(atoi(operand)));
else if(strcmp(opcode,"RESB")==0)
locctr+=(atoi(operand));
else if(strcmp(opcode,"BYTE")==0)
++locctr;
fprintf(fp3,"%s\t%s\t%s\n",label,opcode,operand);
fscanf(fp1,"%s%s%s",label,opcode,operand);
}
fprintf(fp3,"%d\t%s\t%s\t%s\n",locctr,label,opcode,operand);
length=locctr-start;
printf("\nThe length of the program is %d",length);
fclose(fp1);
fclose(fp2);
fclose(fp3);
fclose(fp4);
getch();
}
```

INPUT.DAT:

```
MAIN        START       2000
BEGIN       LDA         NUM1
**          ADD         NUM2
**          LDCH        CHAR1
**          STCH        CHAR2
NUM1        WORD         5
NUM2        RESW        1
CHAR1       BYTE        C'A'
CHAR2       RESB        1
**          END         BEGIN
```

OPTAB.DAT:

```
ADD         18
SUB         1C
MUL         20
DIV         24
LDA         00
LDB         68
LDX         04
LDCH        50
STA         0C
STB         78
STX         10
STCH        54
J           3C
JSUB        48
RSUB        4C
JEQ         30
JLT         38
JGT         34
START       *
END         *
```

SYMTAB.DAT:

```
BEGIN       2000
NUM1        2012
NUM2        2015
CHAR1       2018
CHAR2       2019
```

RESULT:

Thus pass one of a two pass assembler is implemented in C language.

Ex.No: 03

PASS TWO OF TWO PASS ASSEMBLER

AIM:

To implement pass two of a two pass assembler in C language.

ALGORITHM:

1. Open and read the first line from the intermediate file.
2. If the first line contains the opcode "START", then write the label, opcode and operand field values of the corresponding statement directly to the final output file.
3. Do the following steps, until an "END" statement is reached.
   - 3.1 Start writing the location counter, opcode and operand fields of the corresponding statement to the output file, along with the object code.
   - 3.2 If there is no symbol/label in the operand field, then the operand address is assigned as zero and it is assembled with the object code of the instruction
   - 3.3 If the opcode is BYTE, WORD, RESB etc convert the constants to the object code.
4. Close the files and exit

PROGRAM:

```
/*      C Program to implement PASS TWO OF TWO PASS ASSEMBLER      */
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
void main()
{
char a[10],ad[10],label[10],opcode[10],operand[10],mnemonic[10],symbol[10];
int i,locctr,code,add,len,actual_len;
FILE *fp1,*fp2,*fp3,*fp4;
clrscr();
fp1=fopen("twoout.dat","w");
fp2=fopen("symtab.dat","r");
fp3=fopen("out.dat","r");
fp4=fopen("optab.dat","r");
fscanf(fp3,"%s%s%s",label,opcode,operand);
if(strcmp(opcode,"START")==0)
{
fprintf(fp1,"\t%s\t%s\t%s\n",label,opcode,operand);
fscanf(fp3,"%d%s%s%s",&locctr,label,opcode,operand);
}
while(strcmp(opcode,"END")!=0)
{
if(strcmp(opcode,"BYTE")==0)
{
fprintf(fp1,"%d\t%s\t%s\t%s\t",locctr,label,opcode,operand);
len=strlen(operand);
```

```
actual_len=len-3;
for(i=2;i<(actual_len+2);i++)
{
itoa(operand[i],ad,16);
fprintf(fp1,"%s",ad);
}
fprintf(fp1,"\n");
}
else if(strcmp(opcode,"WORD")==0)
{
len=strlen(operand);
itoa(atoi(operand),a,10);
fprintf(fp1,"%d\t%s\t%s\t%s\t00000%s\n",locctr,label,opcode,operand,a);
}
else if((strcmp(opcode,"RESB")==0)||(strcmp(opcode,"RESW")==0))
{
fprintf(fp1,"%d\t%s\t%s\t%s\n",locctr,label,opcode,operand);
}
else
{
rewind(fp4);
fscanf(fp4,"%s%d",mnemonic,&code);
while(strcmp(opcode,mnemonic)!=0)
fscanf(fp4,"%s%d",mnemonic,&code);
if(strcmp(operand,"**")==0)
{
fprintf(fp1,"%d\t%s\t%s\t%s\t%d0000\n",locctr,label,opcode,operand,code);
}
else
{
rewind(fp2);
fscanf(fp2,"%s%d",symbol,&add);
while(strcmp(operand,symbol)!=0)
{
fscanf(fp2,"%s%d",symbol,&add);
}
fprintf(fp1,"%d\t%s\t%s\t%s\t%d%d\n",locctr,label,opcode,operand,code,add);
}
}
fscanf(fp3,"%d%s%s%s\n",&locctr,label,opcode,operand);
}
fprintf(fp1,"%d\t%s\t%s\t%s\n",locctr,label,opcode,operand);
printf("FINISHED");
fclose(fp1);
fclose(fp2);
fclose(fp3);
fclose(fp4);
getch();
}
```

INPUT.DAT:

```
MAIN    START   2000
BEGIN   LDA     NUM1
**      ADD     NUM2
**      LDCH    CHAR1
**      STCH    CHAR2
NUM1    WORD     5
NUM2    RESW    1
CHAR1   BYTE    C'A'
CHAR2   RESB    1
**      END     BEGIN
```

OPTAB:

```
ADD     18
ADDR    90
SUB     1C
SUBR    94
MUL     20
MULR    98
DIV     24
DIVR    9C
LDA     00
LDB     68
LDX     04
LDCH    50
STA     0C
STB     78
STX     10
STCH    54
TIX     2C
J       3C
JSUB    48
RSUB    4C
JEQ     30
JLT     38
JGT     34
START   *
END     *
```

SYMTAB.DAT:

```
BEGIN   2000
NUM1    2012
NUM2    2015
CHAR1   2018
CHAR2   2019
```

RESULT:

Thus pass two of a two pass assembler is implemented in C language.

Ex.No: 04

SINGLE PASS ASSEMBLER

AIM:

To implement a single pass assembler in C language.

ALGORITHM:

1. Open and Read the input file
2. If the input line has the opcode "START" do the following
    - 2.1 Find if there is any operand field after "START", initialize the LC to the operand value
    - 2.2 Otherwise if there is no value in the operand field then LC is set to 0
3. Write the input line to the intermediate file
4. Do the following steps until the opcode is END
    - 4.1 Check the Symbol table, if the symbol is not available then enter that symbol into the SYMTAB, along with the memory address in which it is stored. Otherwise, the error message should be displayed
    - 4.2 If there is a opcode
        - 4.2.1 If opcode is present in the OPTAB, then increment the LC by 3 and Start writing the location counter, opcode and operand fields of the corresponding statement to the output file, along with the object code.
        - 4.2.2 If opcode is "WORD", then increment LC by 3;
        - 4.2.3 If opcode is "BYTE", then increment LC by 1;
        - 4.2.4 If opcode is "RESW" then increment LC by the integer equivalent of the operand value * 3;
        - 4.2.5 If opcode is "RESB", then increment LC by the integer equivalent of the operand value
        - 4.2.6 If there is no symbol/label in the operand field, then the operand address is assigned as zero and it is assembled with the object code of the instruction
        - 4.2.7 Write the processed lines in the intermediate file along with their location counters
    - 5. To find the length of the program, Subtract the starting address of the program from the final value of the LC
6. Close all the files and exit

PROGRAM:

```
/*                          SINGLE PASS ASSEMBLER                          */
#include<stdio.h>
#include<string.h>
#define q 11//no. of mnemonics in the array A
void main()
{
```

```
int lc,ad,address,err=0;
int s,num,l,i=0,j,n=0,line=1,f=0,f1=0,t=0,ni=0,m=0,t1;
FILE *fp1,*fp2,*fp3,*fp4;
char lab[10],op[10],val[10],code[10];
char a[20]
[15]={"STA","STL","LDA","LDB","J","JEQ","J","SUB","COMP","STCH","ADD","SUB"};
char b[20][15]={"14","32","03","69","34","30","48","28","24","16","0C"};
char sym[15][10];
int symadd[15];
clrscr();
fp1=fopen("INPUT.DAT","r");
fp2=fopen("OBJFILE.DAT","w");
fp3=fopen("ERROR.DAT","w");
fp4=fopen("SYMTAB.DAT","w");
while((!feof(fp1)))
{
fscanf(fp1,"%s\t%s\t%s",lab,op,val);
t++;
m++;
if(strcmp(op,".")==0)
m=0;
else if(strcmp(op,"END")==0)
break;
}
t=t-1;
m--;
fclose(fp1);
fp1=fopen("INPUT.DAT","r");
fscanf(fp1,"%s\t%s\t%x",lab,op,&lc);
fprintf(fp3,"-------------------------------------\n");
fprintf(fp3,"LINE NO.\t|ERROR FOUND\n");
fprintf(fp3,"-------------------------------------");
fprintf(fp4,"SYMBOL\tADDRESS");
s=lc;
fprintf(fp2,"H^%s^00%x^%x\n",lab,lc,t*3);
fprintf(fp2,"T^00%x^",lc);
if(m>10)
fprintf(fp2,"1E");
else
fprintf(fp2,"%x",m*3);
        while((op,".")!=0&&(!feof(fp1)))
        {
                fscanf(fp1,"%s\t%s\t%s",lab,op,val);
                line++;
                if(strcmp(lab,"$")!=0)
                        {
                        for(i=0;i<n;i++)
                                {
                                if(strcmp(lab,sym[i])==0)
                                        {
                                                f=1;
```

```
                                        break;
                                    }
                                f=0;
                                }
                        if(f==0)
                        {
                        strcpy(sym[n],lab);
                        symadd[n]=lc;
                        fprintf(fp4,"\n%s\t%x",lab,lc);
                        n++;
                        }
                        if(f==1){
                        fprintf(fp3,"%d\t\t|SYMBOL ALREADY DEFINED\n",line);err++;}
                        }
                num=atoi(val);
                if(strcmp(op,"RESW")==0)
                lc=lc+(num*3);
                else if(strcmp(op,"RESB")==0)
                lc=lc+num;
                else if(strcmp(op,"BYTE")==0)
                        {
                        num=strlen(val)-3;
                        lc=lc+num;
                        for(i=2,j=0;i<strlen(val)-1;i++)
                                {
                                code[j]=val[i];
                                j++;
                                }
                        code[j]='\0';
                        fprintf(fp2,"^%s",code);
                        ni++;
                        }
                else
                        lc=lc+3;
                if(strcmp(op,".")==0)
                break;
        }
        while(strcmp(op,"END")!=0&&(!feof(fp1)))
                {
                fscanf(fp1,"%s\t%s\t%s",lab,op,val);
                line++;
                if(strcmp(op,"END")==0)
                break;
                if((strcmp(lab,"$")!=0)&&((strcmp(op,"RESW")!=0||strcmp(op,"RESB")!=0||
strcmp(op,"WORD")!=0||strcmp(op,"BYTE")==0)))
                        {
                        for(i=0;i<n;i++)
                                {
                                if(strcmp(lab,sym[i])==0)
                                        {
                                        f=1;
```

```
                                        break;
                                        }
                                f=0;
                                }
                        if(f==0)
                        {
                        strcpy(sym[n],lab);
                        symadd[n]=lc;
                        fprintf(fp4,"\n%s\t%x",lab,lc);
                        n++;
                        }
                        else{
                        fprintf(fp3,"\n%d\t\t|SYMBOL ALREADY DEFINED");err++;}
                        }
                else if(strcmp(op,"RESW")==0||strcmp(op,"RESB")==0||
strcmp(op,"WORD")==0||strcmp(op,"BYTE")==0)
                        fprintf(fp3,"\n%d\t\t|Declaration not allowed here",line);
                if(strcmp(op,"RESW")!=0&&strcmp(op,"RESB")!=0&&strcmp(op,"WORD")!
=0&&strcmp(op,"BYTE")!=0)
                        {
                        for(i=0;i<q;i++)
                                {
                                if(strcmp(op,a[i])==0)
                                        {
                                        strcpy(code,b[i]);
                                        f1=0;
                                        break;
                                        }
                                f1=1;
                                }
                        if(f1==1){
                        fprintf(fp3,"\n%d\t\t|WRONG OPCODE",line);err++;}
                        for(i=0;i<n;i++)
                                {
                                if(strcmp(val,sym[i])==0)
                                        {
                                        address=symadd[i];
                                        f=0;
                                        break;
                                        }
                                f=1;
                                }
                        if(f){
                        fprintf(fp3,"\n%d\t\t|UNDEFINED SYMBOL",line);err++;}
                        }
                if(ni<10)
                {
                fprintf(fp2,"^%s%x",code,address);
                ni++;
                }
                else
```

```
                {
                fprintf(fp2,"T^00%x^",lc);
                        if(m>10)
                        {
                        fprintf(fp2,"1E");
                        m=m-10;
                        }
                        else
                        {
                        fprintf(fp2,"%x",m*3);
                        fprintf(fp2,"^%s%x",code,address);
                        ni=0;
                        }
                }
                lc=lc+3;
                }
        fprintf(fp2,"\nE^00%x",s);
        fprintf(fp3,"No of errors=%d\n---------------------------------------",err);
        printf("Output file:OBJCODE.DAT\nErrors are described in ERROR.DAT\nSymbol table
is in the file:SYMTAB.DAT");
        getch();
        fcloseall();
}
```

INPUT.DAT

```
COPY START      1000
RETADR      RESW 1
BUFFER      RESB  4
EOF    BYTE C`EOF`
$       .       $
FIRST STA   RETADR
$       STL     BUFFER
$       J       FIRST
$       END     START
```

RESULT:

Thus single pass assembler is implemented in C language.

Ex.No: 05

TWO PASS MACRO PROCESSOR

AIM:

To implement two pass macro processor in C language.

ALGORITHM:

PROGRAM:

```
/*      program to Perform pass2 of macro   */

#include <stdio.h>

char optable[50][20];
char argtab[20][10];
int cnt = 0;
int def_cnt = 0,nam_cnt = 0,arg_cnt = 0;
FILE *exp;

struct definition
{
        char instruction[20];
        char operand[30];
}deftab[30];

struct name
{
        char MacroName[20];
        int beg;
        int end;
}namtab[5];


void initialize()
{
        FILE *optab,*deftable,*namtable;
        char mnemonic[20],opcode[20];

        optab = fopen("optab.txt","r");
        deftable = fopen("deftab.txt","r");
        namtable = fopen("namtab.txt","r");

        do
        {
                fscanf(optab,"%s %s",mnemonic,opcode);
                strcpy(optable[cnt++],mnemonic);
```

```
	}while(!feof(optab));

	do
	{
		fscanf(deftable,"%s %s",deftab[def_cnt].instruction,deftab[def_cnt].operand);
		def_cnt++;
	}while(!feof(deftable));

	do
	{
		fscanf(namtable,"%s %d
%d",namtab[nam_cnt].MacroName,&namtab[nam_cnt].beg,&namtab[nam_cnt].end);
		nam_cnt++;
	}while(!feof(namtable));

	fclose(deftable);
	fclose(optab);
	fclose(namtable);
}


int ismacro(char *str)
{
	int i;

	for(i=0;i<nam_cnt;i++)
		if(!strcmp(namtab[i].MacroName,str))
			return 1;
	return 0;
}

int iskeyword(char *str)
{
	int i;

	for(i=0;i<cnt;i++)
		if(!strcmp(optable[i],str))
			return 1;
	return 0;
}


void expand(char *name,char *args)
{
	FILE *argtbl;
	int beg,end,i,j,index;
	char operand[30],tmp[20];

	argtbl = fopen("argtab.txt","a+");

	for(i=0;i<nam_cnt;i++)
```

```
{
        if(!strcmp(namtab[i].MacroName,name))
        {
                beg = namtab[i].beg;
                end = namtab[i].end;
        }
}

arg_cnt = 1;

i=0;
do
{
        j=0;
        do
        {
                argtab[arg_cnt][j++] = args[i++];
        }while(args[i] != ',' && args[i] != '\0');

        argtab[arg_cnt][j] = '\0';
        arg_cnt++;

}while(args[i++] != '\0');

fprintf(argtbl,"\n%s :",name);
for(i=0;i<arg_cnt;i++)
{
        fprintf(argtbl,"%s ",argtab[i]);
}


for(i=beg+1;i<=end;i++)
{
        fprintf(exp,"\t%s\t",deftab[i].instruction);

        strcpy(operand,deftab[i].operand);
        for(j=0;j<strlen(operand);j++)
        {
                if(operand[j] == '?')
                {
                        operand[j] = '%';
                        index = operand[j+1]-48;
                        operand[j+1] = 's';
                        sprintf(tmp,operand,argtab[index]);
                        strcpy(operand,tmp);
                }
        }
        fprintf(exp,"%s\n",operand);
}
fclose(argtbl);
getch();
```

```
}


void main()
{
        FILE *source,*argtbl;
        char str[30],str1[30],str2[30];
        int i;

        source = fopen("prog.txt","r");
        argtbl = fopen("argtab.txt","w+");
        exp = fopen("exppgm.txt","w+");
        fclose(argtbl);

        initialize();

        do
        {
                fscanf(source,"%s %s",str,str1);

                beg:
                if(feof(source)){}
                else if(!strcmp(str1,"MACRO"))
                {
                        strcpy(optable[cnt++],str);
                        fscanf(source,"%s",str2);

                        do
                        {
                                fscanf(source,"%s %s",str,str1);
                        }while(strcmp(str,"MEND"));

                        strcpy(str,str1);
                        fscanf(source,"%s",str1);
                        goto beg;
                }
                else if(iskeyword(str))
                {

                        if(ismacro(str))
                        {
                                fprintf(exp,".\t%s\t%s\n",str,str1);
                                expand(str,str1);
                        }
                        else
                                fprintf(exp,"\t%s\t%s\n",str,str1);
                }
                else
                {
                        fscanf(source,"%s",str2);
                        if(ismacro(str1))
```

```
                        {
                                fprintf(exp,".%s\t%s\t%s\n",str,str1,str2);
                                fprintf(exp,"%s",str);
                                expand(str1,str2);
                        }
                        else
                                fprintf(exp,"%s\t%s\t%s\n",str,str1,str2);
                }

        }while(!feof(source));

        fclose(source);
}
```

RESULT:

Thus two pass macro processor is implemented in C language.

Ex.No: 06

SINGLE PASS MACRO PROCESSOR

AIM:

To implement a single pass macro processor in C language.

ALGORITHM:

1. Get the statement from the input file
2. If the statement has the directive “MACRO”, then the number of macro “n” will be incremented by 1
3. Repeat the steps 1 and 2 until an end of file is encountered
4. Open “n” number of macro files in write mode and rewind the input file pointer
5. If the directive is “MACRO” then, do the following
   - 5.1 Enter the macro name present in the operand field
   - 5.2 Write the line to the expanded output file
   - 5.3 Enter the lines in the body of each macro in to the corresponding files already opened in step 4
   - 5.4 Write the body of each macro to the expanded output file until a “MEND” is reached
6. Write the remaining lines directly to the expanded file.

PROGRAM:

```
/*              C Program to implement SINGLE PASS MACRO PROCESSOR          */
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
void main()
{
int n,flag,i;
char ilab[20],iopd[20],oper[20],NAMTAB[20][20];
FILE *fp1,*fp2,*DEFTAB;
clrscr();
fp1=fopen("macroin.dat","r");
fp2=fopen("macroout.dat","w");
n=0;
rewind(fp1);
fscanf(fp1,"%s%s%s",ilab,iopd,oper);
while(!feof(fp1))
{
if(strcmp(iopd,"MACRO")==0)
{
        strcpy(NAMTAB[n],ilab);
        DEFTAB=fopen(NAMTAB[n],"w");
        fscanf(fp1,"%s%s%s",ilab,iopd,oper);
        while(strcmp(iopd,"MEND")!=0)
```

```
        {
                fprintf(DEFTAB,"%s\t%s\t%s\n",ilab,iopd,oper);
                fscanf(fp1,"%s%s%s",ilab,iopd,oper);
        }
        fclose(DEFTAB);
        n++;
}
else
{
        flag=0;
        for(i=0;i<n;i++)
        {
                if(strcmp(iopd,NAMTAB[i])==0)
                {
                        flag=1;
                        DEFTAB=fopen(NAMTAB[i],"r");
                        fscanf(DEFTAB,"%s%s%s\n",ilab,iopd,oper);
                        while(!feof(DEFTAB))
                        {
                                fprintf(fp2,"%s\t%s\t%s\n",ilab,iopd,oper);
                                fscanf(DEFTAB,"%s%s%s",ilab,iopd,oper);
                        }
                        break;
                }
        }
        if(flag==0)
        fprintf(fp2,"%s\t%s\t%s\n",ilab,iopd,oper);
        }
        fscanf(fp1,"%s%s%s",ilab,iopd,oper);
        }
        fprintf(fp2,"%s\t%s\t%s\n",ilab,iopd,oper);
        getch();
}
```

MACROIN.DAT

```
M1      MACRO       **
**      LDA         N1
**      ADD         N2
**      STA         N3
**      MEND        **
M2      MACRO       **
**      LDA         N1
**      SUB         N2
**      STA         N4
**      MEND        **
M3      MACRO       **
**      LDA         N1
**      MUL         N2
**      STA         N5
**      MEND        **
**      START       1000
**      M3          **
**      M2          **
**      M1          **
**      END         **
```

RESULT:

Thus a single pass macro processor is implemented in C language.

Ex.No: 07

ABSOLUTE LOADER

AIM:

To implement an Absolute loader in C language.

ALGORITHM:

1. Read the Header record
2. Verify program name and length
3. Read first Text record from the input file
4. Process the following steps until an End record is reached

5.1 If object code is in character form, convert it to internal hexadecimal representation

5.2 Move object codes to specified locations in memory

5.3 Write the starting location counter value of a block of object code and the corresponding internal representation to the output file

5.4 Read next Text record from the input file

5. Go to the address specified in End record
6. Close all the files and exit

PROGRAM:

```
/*                      C Program to implement ABSOLUTE LOADER                  */
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        char input[10];
        int start,length,address;
        FILE *fp1,*fp2;
        clrscr();
        fp1=fopen("input.dat","r");
        fp2=fopen("output.dat","w");
        fscanf(fp1,"%s",input);
        while(strcmp(input,"E")!=0)
        {
                if(strcmp(input,"H")==0)
                {
                        fscanf(fp1,"%d",&start);
                        fscanf(fp1,"%d",&length);
                        fscanf(fp1,"%s",input);
                }
                else if(strcmp(input,"T")==0)
                {
                        fscanf(fp1,"%d",&address);
                        fscanf(fp1,"%s",input);
```

```
                        fprintf(fp2,"%d\t%c%c\n",address,input[0],input[1]);
                        fprintf(fp2,"%d\t%c%c\n",(address+1),input[2],input[3]);
                        fprintf(fp2,"%d\t%c%c\n",(address+2),input[4],input[5]);
                        address+=3;
                        fscanf(fp1,"%s",input);
                }
                else
                {
                        fprintf(fp2,"%d\t%c%c\n",address,input[0],input[1]);
                        fprintf(fp2,"%d\t%c%c\n",(address+1),input[2],input[3]);
                        fprintf(fp2,"%d\t%c%c\n",(address+2),input[4],input[5]);
                        address+=3;
                        fscanf(fp1,"%s",input);
                }
        }
        fclose(fp1);
        fclose(fp2);
        printf("FINISHED");
        getch();
}
```

INPUT.DAT:

```
H 1000 232
T 1000 142033 483039 102036
T 2000 298300 230000 282030 302015
E
```

RESULT:

Thus an Absolute loader is implemented in C language.

Ex.No: 08

RELOCATING LOADER

AIM:

To implement a Relocating loader in C language.

ALGORITHM:

1. Enter the new starting location to which the object code has to be relocated
2. Read the content of the input file as strings one at a time
3. Transfer the strings from input file to output file, until 'T' is encountered
4. Move the consecutive next three strings from input to output
5. Convert the current string, which is the relocation bit associated with each text record to binary form
6. Make the necessary changes in the corresponding words of object code by adding the new starting address with the address part of the object code for which the corresponding relocation bit is set and store the updated object code in the output
7. Move the object code for which the corresponding relocation bit is not set from output to input without change
8. Repeat steps from 2 to 7 until end record is encountered
9. If object code is in character form, convert it to internal hexadecimal representation
10. Move object codes to specified locations in memory
11. Write the starting location counter value of a block of object code and the corresponding internal hexadecimal representations to the output file

PROGRAM:

```
/*                      C Program to implement RELOCATING LOADER                */
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
void main()
{
        char add[6],length[10],input[10],bitmask[12],binary[12],relocbit;
        int start,len,inp,i,address,opcode,addr,actualadd;
        FILE *fp1,*fp2;
        clrscr();
        printf("Enter the actual starting address: ");
        scanf("%d",&start);
        fp1=fopen("input.dat","r");
        fp2=fopen("output.dat","w");
        fscanf(fp1,"%s",input);
        while(strcmp(input,"E")!=0)
        {
                if(strcmp(input,"H")==0)
                {
                        fscanf(fp1,"%s",add);
                        fscanf(fp1,"%s",length);
                        fscanf(fp1,"%s",input);
```

```
                }
                if(strcmp(input,"T")==0)
                {
                        fscanf(fp1,"%d",&address);
                        fscanf(fp1,"%s",bitmask);
                        address+=start;
                        len=strlen(bitmask);
                        for(i=0;i<len;i++)
                        {
                        fscanf(fp1,"%d",&opcode);
                        fscanf(fp1,"%d",&addr);
                        relocbit=bitmask[i];
                        if(relocbit=='0')
                                actualadd=addr;
                        else
                                actualadd=addr+start;
                        fprintf(fp2,"%d\t%d%d\n",address,opcode,actualadd);
                        address+=3;
                        }
                fscanf(fp1,"%s",input);
                }
        }
        fclose(fp1);
        fclose(fp2);
        printf("FINISHED");
        getch();
}
```

INPUT.DAT:

```
H 1000 200
T 1000 11001 14 1033 48 1039 90 1776 92 1765 57 1765
T 2011 11110 23 1838 43 1979 89 1060 66 1849 99 1477
E 1000
```

Enter the actual starting address: 7000

RESULT:

Thus a Relocating loader is implemented in C language.

Ex.No: 09

PASS ONE OF DIRECT LINKING LOADER

AIM:

To implement pass one of direct-linking loader in C language.

ALGORITHM:

1. Enter the location where the program has to be loaded
2. Assign the address got from the user as the first control section address
3. Read the header record of the control section

a. From the details of the header read and store the control section length in a variable

b. Enter the control section name with its address into the external symbol table

4. For each symbol in the subsequent 'D' records the symbol must be entered into the symbol table along with its address, added along with the corresponding control section until the END record is reached
5. Assign the starting address of next control section as the address of the current control section plus the length of the control section
6. Repeat the process from step 3 to 5 until there is no more records

PROGRAM:

```
/*      C Program to implement PASS ONE OF DIRECT LINKING LOADER          */
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
struct estab
{
char csect[10];
char sym_name[10];
int add,length;
}table[10];
void main()
{
char input[10];
int i,count=0,start,length,loc;
FILE *fp1,*fp2;
clrscr();
fp1=fopen("linkin.dat","r");
fp2=fopen("linkout.dat","w");
printf("\nEnter the location where the program has to be located: ");
scanf("%x",&start);
fprintf(fp2,"CSect\tSym_Name\tAddress\t\tLength\n\n");
rewind(fp1);
while(!feof(fp1))
```

```
{
fscanf(fp1,"%s",input);
if(strcmp(input,"H")==0)
{
fscanf(fp1,"%s",input);
strcpy(table[count].csect,input);
strcpy(table[count].sym_name,"**");
fscanf(fp1,"%s",input);
table[count].add=atoi(input)+start;
fscanf(fp1,"%x",&length);
table[count++].length=length;
fscanf(fp1,"%s",input);
}
if(strcmp(input,"D")==0)
{
fscanf(fp1,"%s%x",input,&loc);
while(strcmp(input,"R")!=0)
{
strcpy(table[count].csect,"**");
strcpy(table[count].sym_name,input);
table[count].add=loc+start;
table[count++].length=0;
fscanf(fp1,"%s%x",input,&loc);
}
while(strcmp(input,"T")!=0)
fscanf(fp1,"%s",input);
}
if(strcmp(input,"T")==0)
while(strcmp(input,"E")!=0)
fscanf(fp1,"%s",input);
fscanf(fp1,"%s",input);
start=start+length;
}
for(i=0;i<count;i++)
fprintf(fp2,"%s\t%s\t\t%x\t\t%x\n",table[i].csect,table[i].sym_name,table[i].add,table[i].length);
fcloseall();
getch();
}
```

OUTPUT:

Enter the location where the program has to be located: 5075

LINKIN.DAT

```
H       PROGA       000000      000070
D       LISTA       000040      ENDA        000054
R       LISTB       ENDB        LISTC       ENDC
T       000020      10          032010      77100004    15001
T       000054      16          100014      15100006    00002F      100014
```

```
M       000024      05          +LISTB
M       000054      06          +LISTC
M       000058      06          +ENDC
M       000064      06          +ENDC
E       000000


H       PROGB       000000      000088
D       LISTB       000060      ENDB        000070
R       LISTA       ENDA        LISTC       ENDC
T       000036      11          03100000    772027      0510030
T       000070      18          100000      05100006    0510020     0510030
M       000037      05          +LISTA
M       000044      05          +ENDA
M       000070      06          +ENDA
M       000074      06          +ENDC
M       000078      06          +ENDC
M       000082      06          +ENDA
E       000000

H       PROGC       000000      000057
D       LISTC       000030      ENDC        000042
R       LISTA       ENDA        LISTB       ENDB
T       000018      12          03100000    77100004    05100000
T       000042      15          100030      100008      100011      100000
M       000019      05          +LISTA
M       000023      05          +LISTB
M       000027      05          +ENDA
M       000048      06          +LISTA
M       000051      06          +ENDA
M       000054      06          +LISTB
E       000000
```

LINKOUT.DAT

CSect	Sym_Name	Address	Length
PROGA	**	5075	70
**	LISTA	50b5	0
**	ENDA	50c9	0
PROGB	**	50e5	88
**	LISTB	5145	0
**	ENDB	5155	0
PROGC	**	516d	57
**	LISTC	519d	0
**	ENDC	51af	0

RESULT:

Thus pass one of direct-linking loader is implemented in C language.

Ex.No: 10

PASS TWO OF DIRECT LINKING LOADER

AIM:

To implement pass two of direct-linking loader in C language.

ALGORITHM:

1. Enter the location where the program has to be loaded
2. Assign the address got from the user as the first control section address
3. Read the header record of the control section
    i. From the details of the header read and store the control section length in a variable
    ii. Enter the control section name with its address into the external symbol table
4. For each symbol in the subsequent 'D' records the symbol must be entered into the symbol table along with its address, added along with the corresponding control section until the END record is reached
5. Assign the starting address of next control section as the address of the current control section plus the length of the control section
6. Repeat the process from step 3 to 5 until there is no more records

PROGRAM:

```
/*      C Program to implement PASS TWO OF DIRECT LINKING LOADER          */
#include<stdio.h>
#include<conio.h>
main()
{
FILE *fp1,*fp2,*fp3,*fp4,*fp5,*fp6,*fp7,*fp8;
char rec[20],name[20],len[20],t[20],string[1000],mod[40][40],est1[20],est2[20],est4[20],sign[40]
[1],temp;
int ptn1[20][20],ptn2[20][20];
int
l,h=0,lent[20],i,st,m1,m2,start[20],add[20],k=0,est3,z1=0,val[40],ptn[40],offset[40],j,num,progsta
rt,count=0;
fp1=fopen("DLL_IN.txt","r");
fp2=fopen("ESTAB.txt","r");
fp3=fopen("ntemp.txt","w");
fp4=fopen("memory.txt","w");
fp6=fopen("six.txt","w");
fscanf(fp1,"%s%s%x%s",rec,name,&st,len);
for(l=0;l<3;l++)
{
if(l==1)
fp3=fopen("ntempb.txt","w");
if(l==2)
fp3=fopen("ntempc.txt","w");
```

```
fscanf(fp1,"%s",t);
do{
if(strcmp(t,"T")==0)
{
fscanf(fp1,"%x%x",&start[h],&lent[h]);
if(h!=0) {
for(i=0;i<(start[h]-(start[h-1]+lent[h-1]));i++)
fprintf(fp3,"x");
}
h++;
fscanf(fp1,"%s",t);
do{
fprintf(fp3,"%s",t);
fscanf(fp1,"%s",t);
}while((strcmp(t,"T")!=0)&&(strcmp(t,"M")!=0));
}
else if(strcmp(t,"M")==0)
{
fscanf(fp1,"%x%x%s%s",&ptn[k],&offset[k],sign[k],mod[k]);
fscanf(fp2,"%s%s%x%s",est1,est2,&est3,est4);
progstart=est3;
do{
if(strcmp(mod[k],est2)==0){
val[z1]=est3;
z1++;
break;
}
else if(strcmp(mod[k],est1)==0) {
val[z1]=est3;
z1++;
break;
}
fscanf(fp2,"%s%s%x%s",est1,est2,&est3,est4);
}while(!feof(fp2));
rewind(fp2);
fscanf(fp1,"%s",t);
k++;
}
else if(strcmp(t,"E")==0)
{
fscanf(fp1,"%s",t);
if(l!=2)
fscanf(fp1,"%s%s%x%s",rec,name,&st,len);
break;
}
else if(strcmp(t,"R")==0)
{
while(strcmp(t,"T")!=0)
fscanf(fp1,"%s",t);
}
else if(strcmp(t,"D")==0)
```

```
{
while(strcmp(t,"R")!=0)
fscanf(fp1,"%s",t);
}
}while(1);
fclose(fp3);
for(i=0;i<k;i++){
if(l==0)
fp3=fopen("ntemp.txt","r+");
if(l==1)
fp3=fopen("ntempb.txt","r+");
if(l==2)
fp3=fopen("ntempc.txt","r+");
fp5=fopen("temp1.txt","w");
fseek(fp3,(ptn[i]*2)+1,0);
for(j=0;j<offset[i];j++)
{
fscanf(fp3,"%c",&temp);
fprintf(fp5,"%c",temp);
}
fprintf(fp5,"\n");
fclose(fp5);
fp5=fopen("temp1.txt","r");
fscanf(fp5,"%x",&num);
if(sign[i][0]=='+')
{
num=num+val[i];
fseek(fp3,(ptn[i]*2)+1,0);
if(offset[i]==5)
fprintf(fp3,"0%x",num);
else
fprintf(fp3,"00%x",num);
}
else
{
num=num-val[i];
fseek(fp3,(ptn[i]*2)+1,0);
fprintf(fp3,"00%x",num);
}
fclose(fp3);
fclose(fp5);
}
k=0;h=0;z1=0;
}
fp3=fopen("ntemp.txt","r");
fscanf(fp3,"%s",string);
fclose(fp3);
fprintf(fp6,"%s",string);
fp3=fopen("ntempb.txt","r");
fscanf(fp3,"%s",string);
fclose(fp3);
```

```
fprintf(fp6,"%s",string);
fp3=fopen("ntempc.txt","r");
fscanf(fp3,"%s",string);
fclose(fp3);
fprintf(fp6,"%s",string);
fclose(fp6);
fp6=fopen("six.txt","r");
fscanf(fp6,"%s",string);
for(i=0;i<strlen(string);i++)
{
if(i==0) {
fprintf(fp4,"%x\t",progstart);
progstart+=16;
}
if((i%8==0)&&(i!=0))
fprintf(fp4,"\t");
if((i%32==0)&&(i!=0)) {
fprintf(fp4,"\n");
fprintf(fp4,"%x\t",progstart);
progstart+=16;
}
fprintf(fp4,"%c",string[i]);
}
return 0;
}
```

DLL_IN.TXT

```
H PROGA 000000 00003A
D LISTA 000030 ENDA 000050 .
R LISTB LISTC ENDC
T 000000 1D 172027 4B100000 032023 290000 332007 4B100000 3F2FEC 032016 0F2016
T 00001D 0D 010003 0F200A 4B100000 3E2000
M 000004 05 + LISTB
M 000011 05 + LISTC
E 000000
H PROGB 000000 00002E
D LISTB 000060 ENDB 000070 .
R LISTA ENDA
T 000000 1D B410 B400 B440 77201F E3201B 332FFA DB2015 A004 332009 57900000 B850
T 000020 0E 3B2FE9 13100000 4F0000 F1000000
M 000007 05 + LISTA
M 000022 05 + ENDA
E 000000
H PROGC 000000 00001C
D LISTC 000030 ENDC 000042 .
R LISTA ENDA
T 000000 1C B410 77100000 E32012 332FFA 53900000 DF2008 B850 3B2FEE 4F000005
M 000006 05 + LISTA
M 000013 06 + ENDA
E 000000
```

ESTAB.TXT

PROGA - 003000 000063
- LISTA 003030 -
- ENDA 003050 -
PROGB - 003063 00007f
- LISTB 0030c3 -
- ENDB 0030d3 -
PROGC - 0030e2 000051
- LISTC 003112 -
- ENDC 003124 –

MEMORY.TXT

3000	1720274B	1030c303	20232900	00332007
3010	4B103112	3F2FEC03	20160F20	16010003
3020	0F200A4B	1000003E	2000B410	B400B440
3030	77205013	201B332F	FADB2015	A0043320
3040	09579000	00B850xx	x3B2FE91	30305004
3050	F0000F10	00000B41	07710000	0E050423
3060	32FFA539	00000DF2	008B0034	02FEE4F0
3070	00005			

RESULT:

Thus pass two of direct-linking loader is implemented in C language.

Ex.No: 11

SIMPLE TEXT EDITOR

AIM:

To implement simple text editor with features like insertion/deletion of a character, word and sentence in C language.

ALGORITHM:

1. Design a Menu using switch case
2. Get the choice
3. If choice is “1” then, enter the filename in which the text is to be saved and, type the text using editor and type CTRL+Z to terminate
4. If choice is “2” then enter the filename to be open, If the filename is found, open the file in read mode else display not found
5. If choice is “3” then replace the existing file by a new file name
6. If choice is “4” then insert character or word or sentence

PROGRAM:

```
/*                          SIMPLE TEXT EDITOR                          */
#include<stdio.h>
#include<string.h>
#include<conio.h>
#include<stdlib.h>
void newf();
void view();
void saveas();
void modify();
void main()
{
int op;
clrscr();
do
{
printf("\nMENU\n");
printf("1.New\n");
printf("2.Open\n");
printf("3.Save As\n");
printf("4.Modify\n");
printf("5.Exit\n");
printf("Enter u'r choice: ");
scanf("%d",&op);
switch(op)
{
case 1:
```

```
newf();
break;
case 2:
view();
break;
case 3:
saveas();
break;
case 4:
modify();
break;
case 5:
exit(0);
default:
printf("\nWrong choice!!!");
break;
}
}while(op!=5);
}
void newf()
{
FILE *f;
char fname[20],c;
printf("\nEnter the Filename: ");
scanf("%s",fname);
printf("\nType the content and CTRL+Z to terminate:\n");
f=fopen(fname,"w");
rewind(f);
while((c=getchar())!=EOF)
{
putc(c,f);
}
fclose(f);
}
void view()
{
FILE *f;
char fname[20],c;
printf("\nEnter the name of the file:");
scanf("%s",&fname);
if(searchpath(fname))
{
f=fopen(fname,"r");
while((c=getc(f))!=EOF)
{
printf("%c",c);
}
}
else
printf("\nThe file %s does not exist",fname);
fclose(f);
```

```
}
void saveas()
{
FILE *f1,*f2;
char c,sou[20],des[20];
printf("\nEnter the Source file name: ");
scanf("%s",sou);
if(searchpath(sou))
{
printf("Enter the Destination file name: ");
scanf("%s",des);
f1=fopen(sou,"r");
f2=fopen(des,"w");
while((c=getc(f1))!=EOF)
{
putc(c,f2);
}
fclose(f1);
fclose(f2);
}
else
printf("\nFile does not exist");
getch();
}
void modify()
{
int ch1;
FILE *f1;
char c,*word,*sent,fname[20];
printf("Enter the filename to be modified: ");
scanf("%s",fname);
if(searchpath(fname))
{
printf("\n1.Character");
printf("\n2.Word");
printf("\n3.Sentence");
printf("\nEnter U'r choice: ");
scanf("%d",&ch1);
if(ch1==1)
{
f1=fopen(fname,"a+");
fseek(f1, 0L, SEEK_END);
printf("Enter the character and CTRL+Z to exit:\n ");
while((c=getchar())!=EOF)
{
putc(c,f1);
}
}
else if(ch1==2)
{
printf("Enter the word: ");
```

```
scanf("%s",word);
f1=fopen(fname,"a+");
fseek(f1, 0L, SEEK_END);
fputs(word,f1);
}
else
{
printf("Enter the sentence and CTRL+Z to exit: ");
f1=fopen(fname,"a+");
fseek(f1, 0L, SEEK_END);
while((c=getchar())!=EOF)
{
putc(c,f1);
}
}
}
else
printf("\nFilename does not exist");
fcloseall();
}
```

RESULT:

Thus a simple text editor with features like insertion/deletion of a character, word and sentence is implemented in C language.