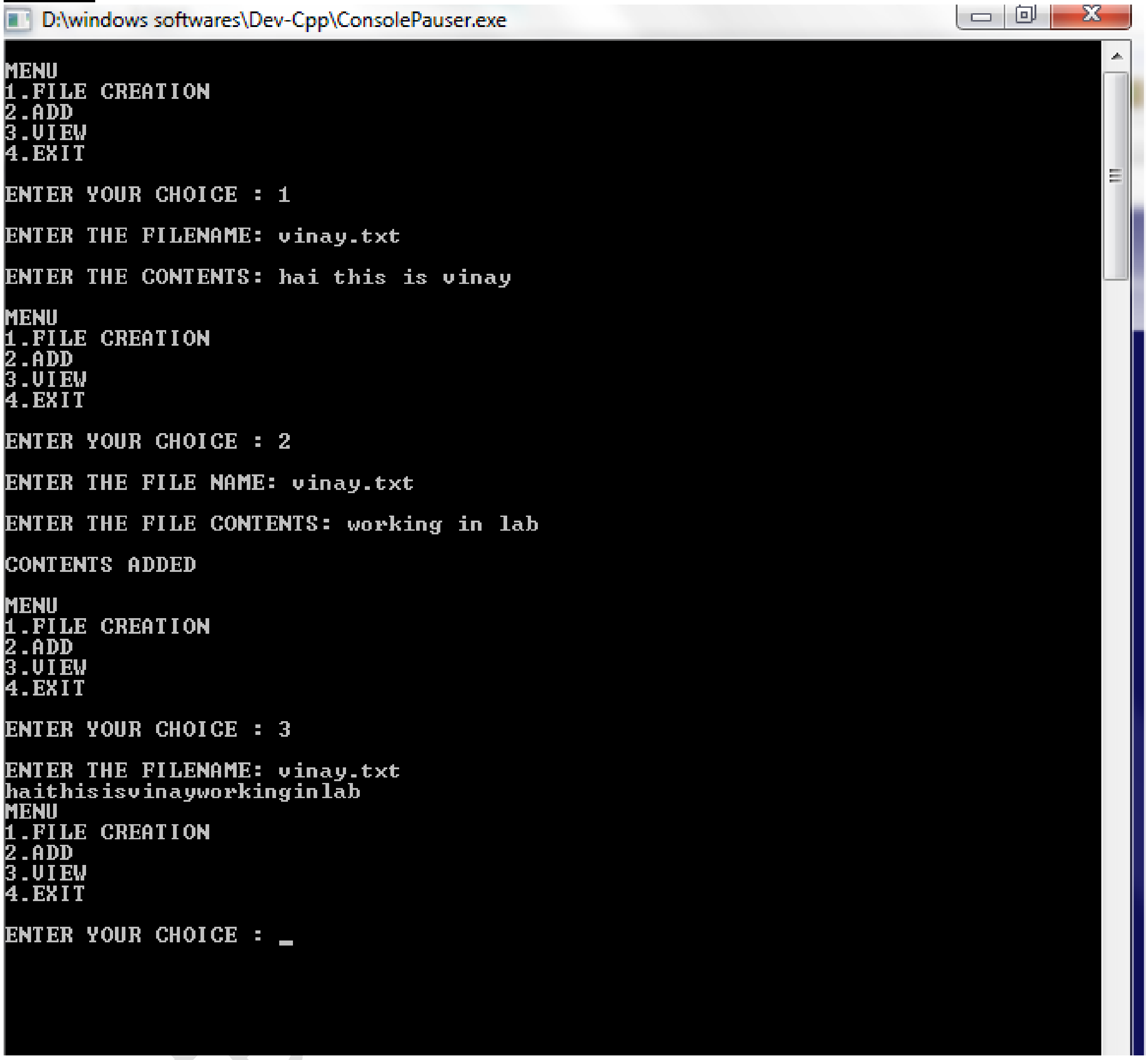
SYSTEMS PROGRAMMING LAB(Prof: prabhakar Rao)

Name: B. Naga Vinay
Regno: 10mse1064

1)**Text editor code**:

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

void menu();
void create();
void add();
void view();
int main()
{
menu();
getch();
}

void menu()
{
int c;
do
{
printf("\nMENU\n");
printf("1.FILE CREATION\n");
printf("2.ADD\n");
printf("3.VIEW\n");
printf("4.EXIT\n");
printf("\nENTER YOUR CHOICE : ");
scanf("%d",&c);
switch(c)
{
case 1:
create();
break;
case 2:
add();
break;
case 3:
view();
break;
case 4:
exit(0);
break;
default:
break;
}
}
while(c!=4);
}

void create()
{
FILE *fp;
char name[20],inp[40];
printf("\nENTER THE FILENAME: ");
scanf("%s",&name);
fp=fopen(name,"w");
printf("\nENTER THE CONTENTS: ");
fflush(stdin);//allows you to flush [clear] the input buffer
gets(inp);
fprintf(fp,"%s",inp);
fclose(fp);
}

void add()
{
FILE *fp;
char name[20],inp[40];
printf("\nENTER THE FILE NAME: ");
scanf("%s",&name);
fp=fopen(name,"a");
if(fp==NULL)
{
printf("\nERROR:file not found\n");
getch();
menu();
}

printf("\nENTER THE FILE CONTENTS: ");
fflush(stdin);//allows you to flush [clear] the input buffer
gets(inp);
fprintf(fp,"%s",inp);
fclose(fp);
printf("\nCONTENTS ADDED\n");
getch();
}

void view()
{
FILE *fp;
char a[15];
char fname[20];
printf("\nENTER THE FILENAME: ");
scanf("%s",&fname);
fp=fopen(fname,"r");
if(fp==NULL)
{
printf("ERROR:file not found\n");
getch();
menu();
}
while((fscanf(fp,"%s",a))!=EOF)
{
printf("%s",a);
//a=fgetc(fp);
}
getch();
fclose(fp);
}
```

output:



```
MENU
1.FILE CREATION
2.ADD
3.VIEW
4.EXIT

ENTER YOUR CHOICE : 1

ENTER THE FILENAME: vinay.txt

ENTER THE CONTENTS: hai this is vinay

MENU
1.FILE CREATION
2.ADD
3.VIEW
4.EXIT

ENTER YOUR CHOICE : 2

ENTER THE FILE NAME: vinay.txt

ENTER THE FILE CONTENTS: working in lab

CONTENTS ADDED

MENU
1.FILE CREATION
2.ADD
3.VIEW
4.EXIT

ENTER YOUR CHOICE : 3

ENTER THE FILENAME: vinay.txt
haithisisvinayworkinginlab
MENU
1.FILE CREATION
2.ADD
3.VIEW
4.EXIT

ENTER YOUR CHOICE : _
```

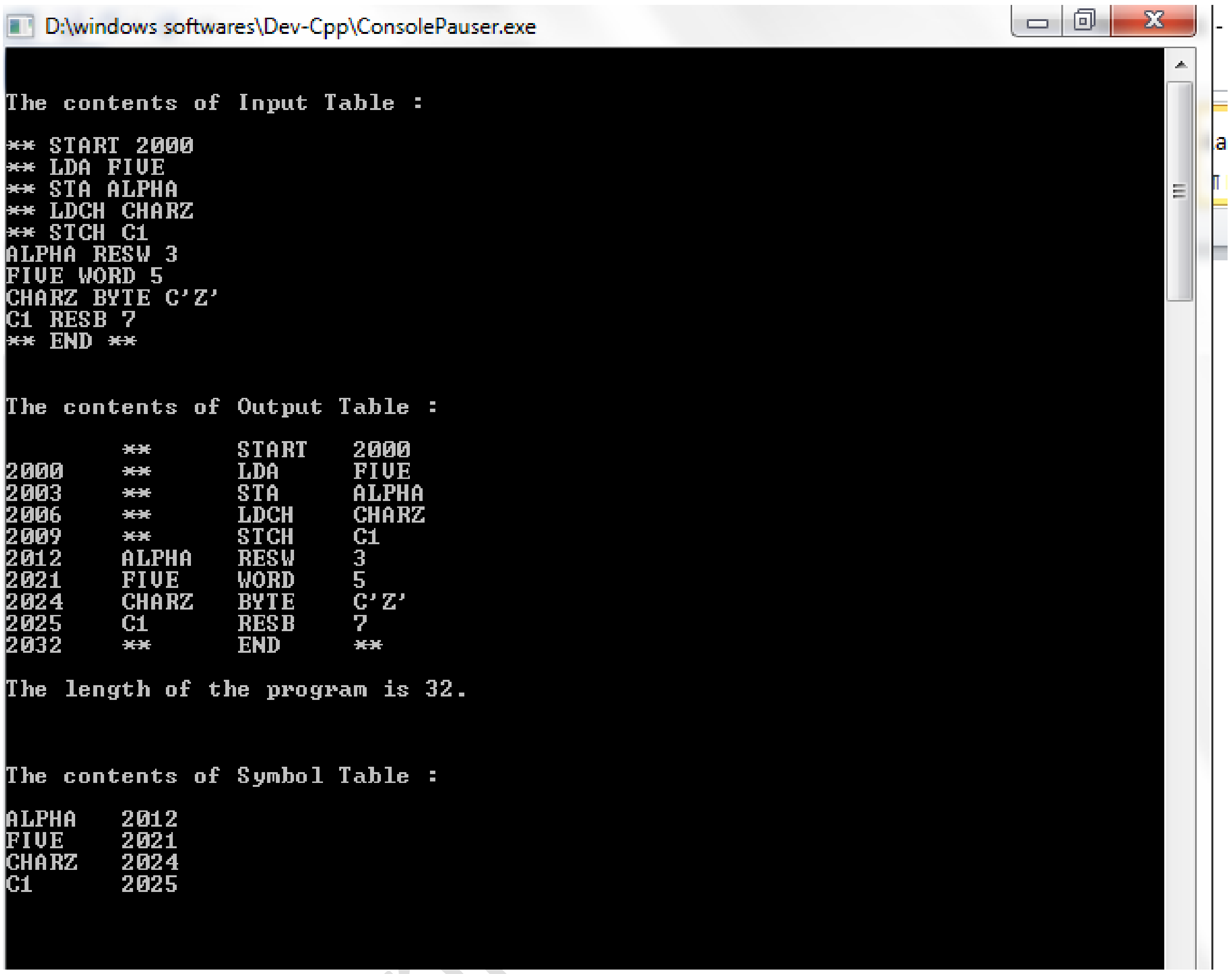
2)pass1 assembler code:

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

main()

{

  char opcode[10],operand[10],label[10],code[10][10],ch;

  char mnemonic[10][10]={"START","LDA","STA","LDCH","STCH","END"};
```

```
int locctr,start,len,i=0,j=0;
FILE *fp1,*fp2,*fp3;


fp1=fopen("INPUT.DAT","r");
fp2=fopen("SYMTAB.DAT","w");
fp3=fopen("OUT.DAT","w");
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",locctr);
  if(strcmp(label,"**")!=0)
  fprintf(fp2,"%s\t%d\n",label,locctr);
  strcpy(code[i],mnemonic[j]);
  while(strcmp(mnemonic[j],"END")!=0)
  {
    if(strcmp(opcode,mnemonic[j])==0)
    {
      locctr+=3;
      break;
    }
```

```
      strcpy(code[i],mnemonic[j]);
      j++;
    }
    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,"\t%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);
fclose(fp1);fclose(fp2);fclose(fp3);
printf("\n\nThe contents of Input Table :\n\n");
fp1=fopen("INPUT.DAT","r");
ch=fgetc(fp1);
while(ch!=EOF)
{
   printf("%c",ch);
   ch=fgetc(fp1);
}
printf("\n\nThe contents of Output Table :\n\n\t");
fp3=fopen("OUT.DAT","r");
ch=fgetc(fp3);
while(ch!=EOF)
{
    printf("%c",ch);

    ch=fgetc(fp3);

  }

  len=locctr-start;

  printf("\nThe length of the program is %d.\n\n",len);

  printf("\n\nThe contents of Symbol Table :\n\n");

  fp2=fopen("SYMTAB.DAT","r");

  ch=fgetc(fp2);

  while(ch!=EOF)

  {

    printf("%c",ch);

    ch=fgetc(fp2);

  }

  fclose(fp1);

  fclose(fp2);

  fclose(fp3);

  getch();

}
```

Output:


D:\windows softwares\Dev-Cpp\ConsolePauser.exe

```
The contents of Input Table :

** START 2000
** LDA FIVE
** STA ALPHA
** LDCH CHARZ
** STCH C1
ALPHA RESW 3
FIVE WORD 5
CHARZ BYTE C'Z'
C1 RESB 7
** END **


The contents of Output Table :

        **      START   2000
2000    **      LDA     FIVE
2003    **      STA     ALPHA
2006    **      LDCH    CHARZ
2009    **      STCH    C1
2012    ALPHA   RESW    3
2021    FIVE    WORD    5
2024    CHARZ   BYTE    C'Z'
2025    C1      RESB    7
2032    **      END     **

The length of the program is 32.


The contents of Symbol Table :

ALPHA   2012
FIVE    2021
CHARZ   2024
C1      2025
```

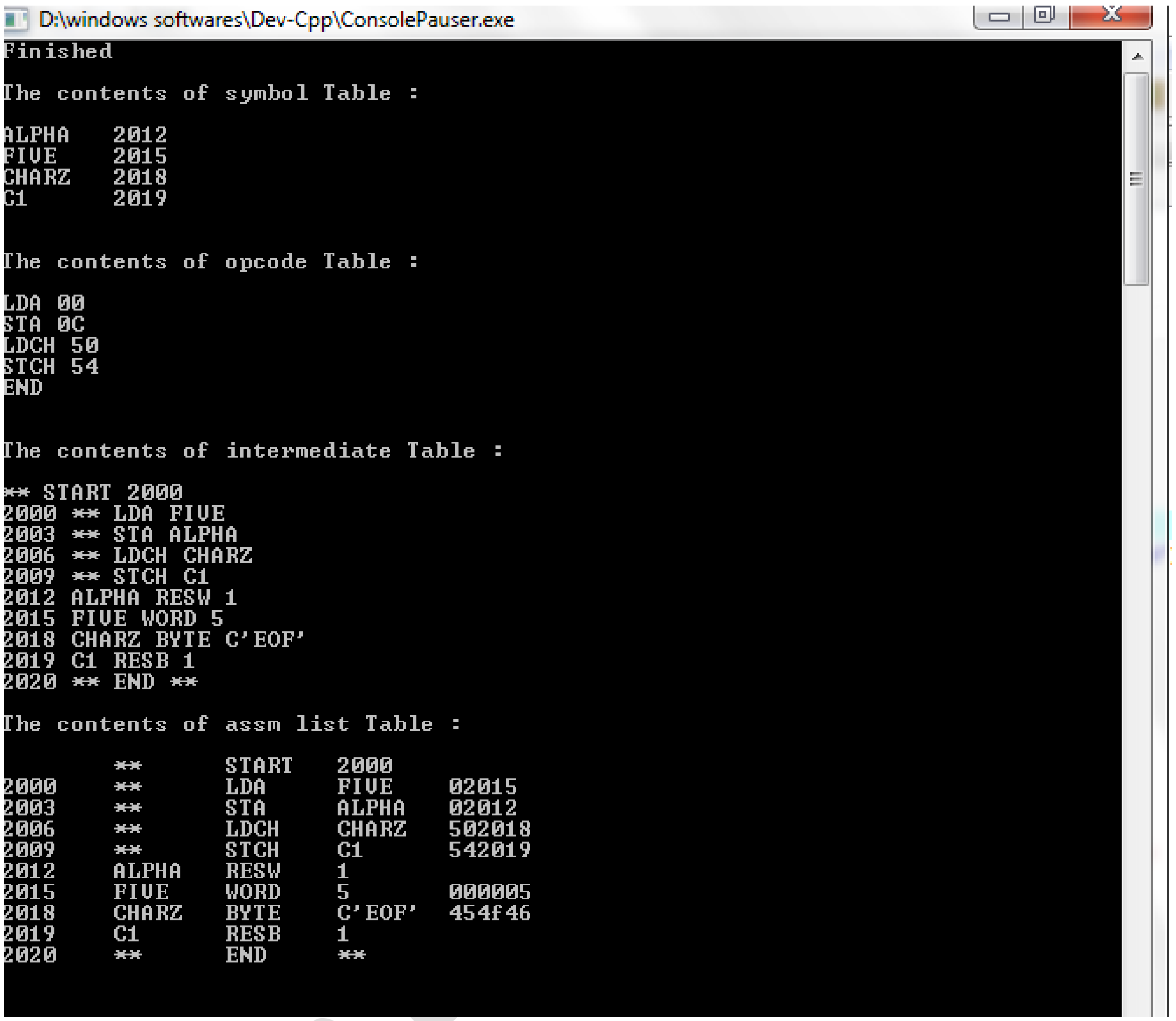
pass2 Assembler:

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<stdlib.h>

main()

{

   char a[10],ad[10],label[10],opcode[10],operand[10],mnemonic[10],symbol[10],ch;

   int i,address,code,add,len,actual_len;

   FILE *fp1,*fp2,*fp3,*fp4;
```

```
fp1=fopen("assmlist.dat","w");

fp2=fopen("symtab.dat","r");

fp3=fopen("intermediate.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",&address,label,opcode,operand);

}

while(strcmp(opcode,"END")!=0)

{

   if(strcmp(opcode,"BYTE")==0)

   {

      fprintf(fp1,"%d\t%s\t%s\t%s\t",address,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",address,label,opcode,operand,a);
}
else if((strcmp(opcode,"RESB")==0)||(strcmp(opcode,"RESW")==0))
{
  fprintf(fp1,"%d\t%s\t%s\t%s\n",address,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",address,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",address,label,opcode,operand,code,add);
  }
}
fscanf(fp3,"%d%s%s%s",&address,label,opcode,operand);
```

```
}
fprintf(fp1,"%d\t%s\t%s\t%s\n",address,label,opcode,operand);
printf("Finished");
fclose(fp1);
fclose(fp2);
fclose(fp3);
fclose(fp4);

printf("\n\nThe contents of symbol Table :\n\n");
fp2=fopen("symtab.dat","r");
ch=fgetc(fp2);
while(ch!=EOF)
{
  printf("%c",ch);
  ch=fgetc(fp2);
}
fclose(fp2);

printf("\n\nThe contents of opcode Table :\n\n");
fp4=fopen("optab.dat","r");
ch=fgetc(fp4);
while(ch!=EOF)
{
  printf("%c",ch);
  ch=fgetc(fp4);
}
fclose(fp4);

printf("\n\nThe contents of intermediate Table :\n\n");
```

```
  fp3=fopen("intermediate.dat","r");

  ch=fgetc(fp3);

  while(ch!=EOF)

  {

    printf("%c",ch);

    ch=fgetc(fp3);

  }

  fclose(fp3);


  printf("\n\nThe contents of assm list Table :\n\n");

  fp1=fopen("assmlist.dat","r");

  ch=fgetc(fp1);

  while(ch!=EOF)

  {

    printf("%c",ch);

    ch=fgetc(fp1);

  }

  fclose(fp1);

  getch();

}
```
OUTPUT:


D:\windows softwares\Dev-Cpp\ConsolePauser.exe

```
Finished

The contents of symbol Table :

ALPHA   2012
FIVE    2015
CHARZ   2018
C1      2019


The contents of opcode Table :

LDA 00
STA 0C
LDCH 50
STCH 54
END


The contents of intermediate Table :

** START 2000
2000 ** LDA FIVE
2003 ** STA ALPHA
2006 ** LDCH CHARZ
2009 ** STCH C1
2012 ALPHA RESW 1
2015 FIVE WORD 5
2018 CHARZ BYTE C'EOF'
2019 C1 RESB 1
2020 ** END **

The contents of assm list Table :

        **      START   2000
2000    **      LDA     FIVE    02015
2003    **      STA     ALPHA   02012
2006    **      LDCH    CHARZ   502018
2009    **      STCH    C1      542019
2012    ALPHA   RESW    1
2015    FIVE    WORD    5       000005
2018    CHARZ   BYTE    C'EOF'  454f46
2019    C1      RESB    1
2020    **      END     **
```

<u>SINGLE PASS ASSEMBLER CODE:</u>

```
#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#define q 11//no. of mnemonics in the array A

int 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;
```

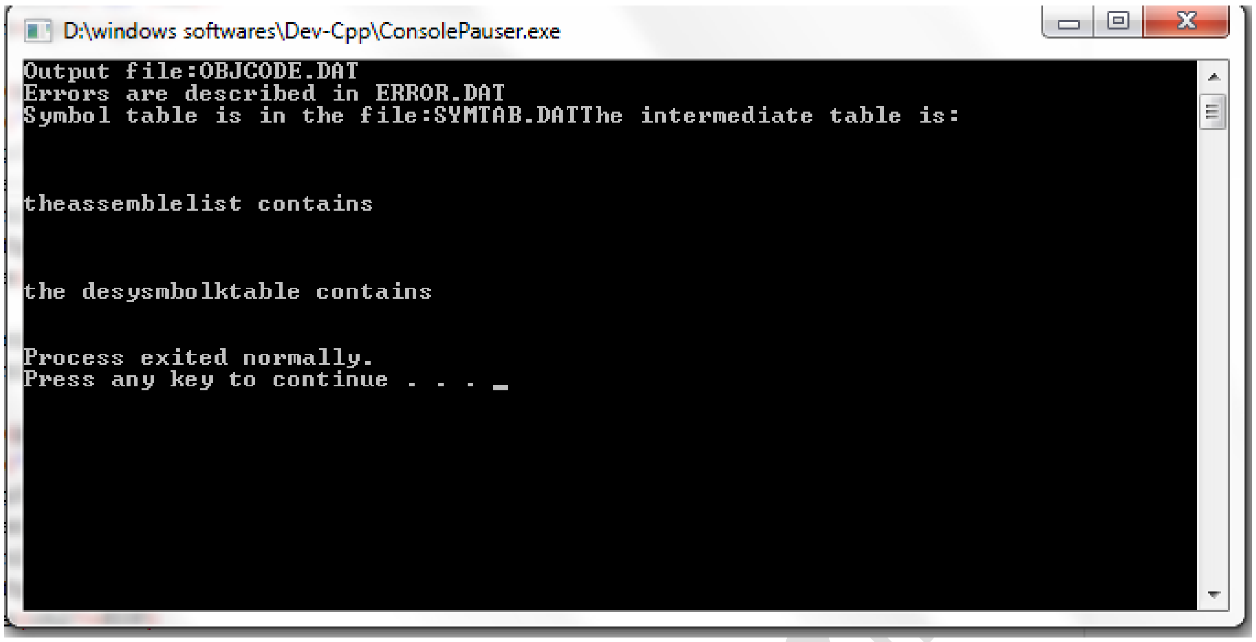
```
fprintf(fp2,"H^%s^00%x^%x\n",lab,lc,t*3);

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,"BYT
E")==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");
  void  fcloseall(int);
getch();
}
```
OUTPUT:



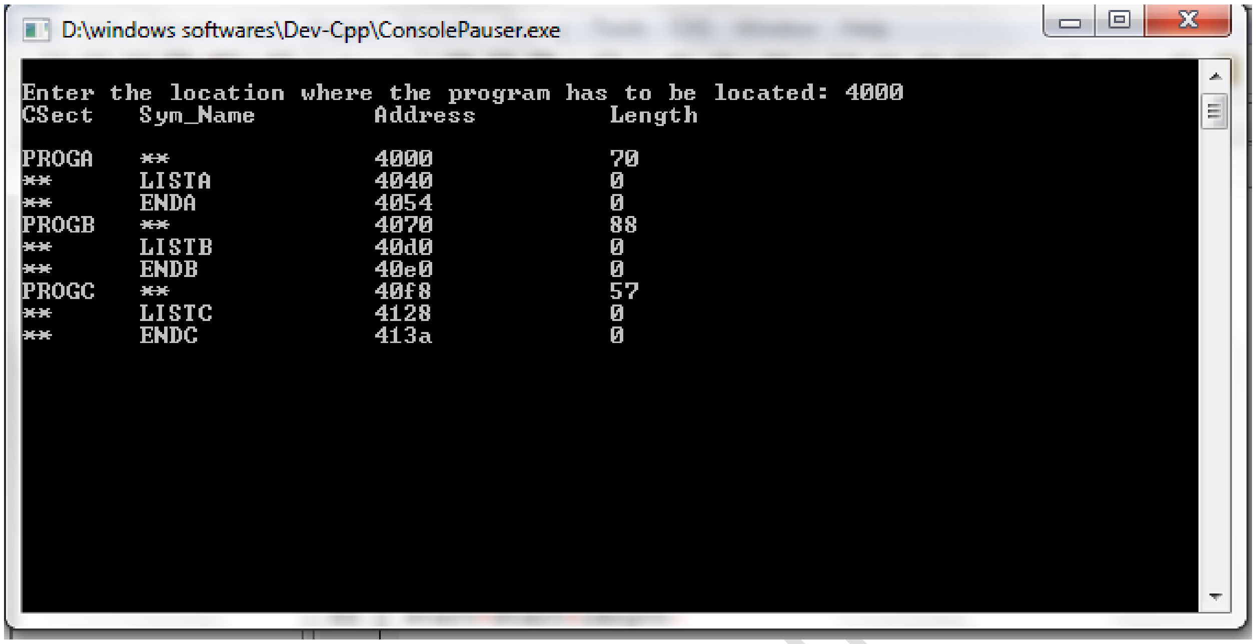
```
Output file:OBJCODE.DAT
Errors are described in ERROR.DAT
Symbol table is in the file:SYMTAB.DATThe intermediate table is:


theassemblelist contains


the desysmbolktable contains


Process exited normally.
Press any key to continue . . .
```

PASS1DIRECTLINKING 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];

int main()

{

char input[10];

int i,count=0,start,length,loc;

FILE *fp1,*fp2;

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();

fclose(fp1);

fclose(fp2);

FILE *p2;

p2=fopen("linkout.dat","r");

char ch1;

 ch1=fgetc(p2);

 while(ch1!=EOF)

 {

    printf("%c",ch1);

    ch1=fgetc(p2);

 }

fclose(p2);

getch();
```

}


D:\windows softwares\Dev-Cpp\ConsolePauser.exe

```
Enter the location where the program has to be located: 4000
CSect   Sym_Name        Address         Length

PROGA   **              4000            70
**      LISTA           4040            0
**      ENDA            4054            0
PROGB   **              4070            88
**      LISTB           40d0            0
**      ENDB            40e0            0
PROGC   **              40f8            57
**      LISTC           4128            0
**      ENDC            413a            0
```

PASS2 DIRECT LINKING LOADER:

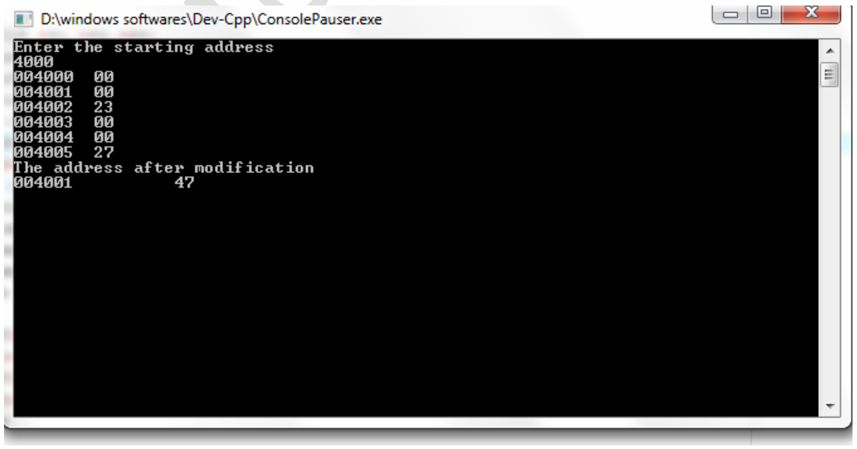
```
#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<stdlib.h>

int main()

{

FILE *f1,*f2,*f3;

int csaddr,progaddr,execaddr,cslen,i,j,k=0,staddr1,staddr2,addr2;

int modadr,val1,adr2,outadr1,esadr;

char outadr[10],adr1[10],name[20],val[10],pname[10],symname[10],adr[10];

char l[10],line[80],len[10],staddr[10],addr[10],addr1[10];

f3=fopen("estab.txt","r");

f2=fopen("dupout.txt","w");

//clrscr();

printf("Enter the starting address\n");

scanf("%d",&progaddr);
```

```
csaddr=progaddr;
execaddr=progaddr;
do
{
if(k==0)
f1=fopen("link2in.txt","r");
if(k==1)
f1=fopen("linking2.txt","r");
do
{
fscanf(f1,"%s",line);
if(line[0]=='H')
{
for(i=9,j=0;i<15,j<6;i++,j++)
addr[j]=line[i];
addr[j]='\0';
for(i=16,j=0;i<20,j<5;i++,j++)
len[j]=line[i];
len[j]='\0';
cslen=atoi(len);
}
else if(line[0]!='E')
{
do
{
fscanf(f1,"%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);

staddr2=staddr1+progaddr;

i=12;

while(line[i]!='$')

{

if(line[i]!='^')

{

printf("00%d\t%c%c\n",staddr2,line[i],line[i+1]);

fprintf(f2,"00%d\t%c%c\n",staddr2,line[i],line[i+1]);

staddr2++;

i=i+2;

}

else

i++;

}

fclose(f2);

}

else if(line[0]=='M')

{

for(i=13,j=0;line[i]!='$',j<5;i++,j++)

name[j]=line[i];

name[j]='\0';

do

{

fscanf(f3,"%s%s%s%s",pname,symname,adr,l);

if(strcmp(name,symname)==0)

{
```

```
for(i=2,j=0;i<8,j<6;i++,j++)

adr1[j]=line[i];

adr1[j]='\0';

adr2=atoi(adr1);

adr2=adr2+progaddr;

f2=fopen("dupout.txt","r");

fscanf(f2,"%s%s",outadr,val);

printf("The address after modification\n");

do

{

outadr1=atoi(outadr);

if(adr2==outadr1)

{

val1=atoi(val);

esadr=atoi(adr);

modadr=val1+esadr;

printf("%s\t\t%d\n",outadr,modadr);

}

fscanf(f2,"%s%s",outadr,val);

}

while(!feof(f2));

}


}while(!feof(f3));

}

}while(line[0]!='E');

}

else

{
```

```
for(i=2,j=0;i<8,j<6;i++,j++)

addr1[j]=line[i];

addr1[j]='\0';

if(strcmp(addr,addr1)==0)

{

addr2=atoi(addr1);

execaddr=csaddr+cslen;

}

else

csaddr=csaddr+cslen;

}

fscanf(f1,"%s",line);

}while(!feof(f1));

k++;

}while(k<=2);

fclose(f1);

fclose(f2);

fclose(f3);

printf("The exec addr is %d",execaddr);

getch();

}
```



```
Enter the starting address
4000
004000  00
004001  00
004002  23
004003  00
004004  00
004005  27
The address after modification
004001          47
```

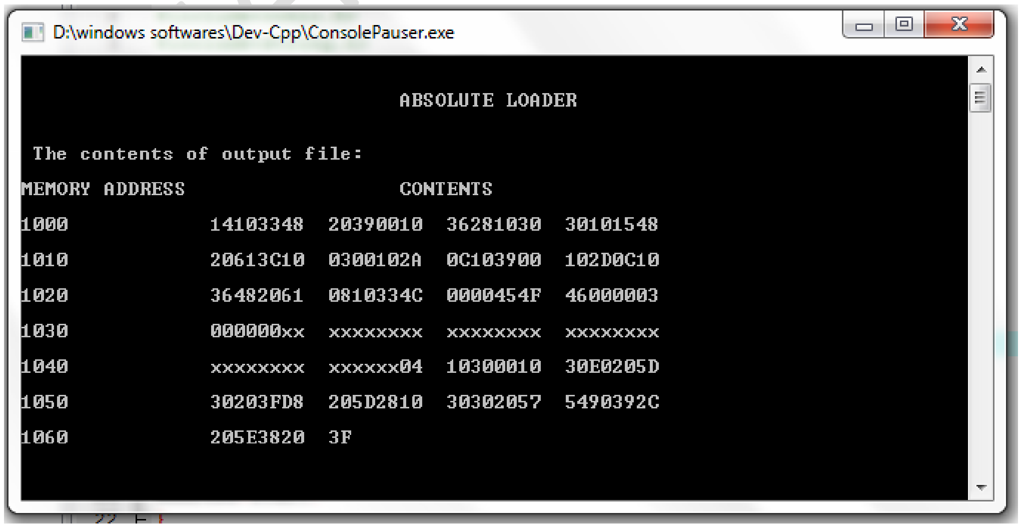
ABSOLUTE LOADER CODE:

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

char input[10],label[10],ch1,ch2;

int addr,w=0,start,ptaddr,l,length=0,end,count=0,k,taddr,address,i=0;

FILE *fp1,*fp2;

void check();

int main()

{



fp1=fopen("INPUT.DAT","r");

fp2=fopen("OUTPUT.dat","w");

fscanf(fp1,"%s",input);

printf("\n\n\t\t\t\tABSOLUTE LOADER\n");

fprintf(fp2,"MEMORY ADDRESS\t\t\tCONTENTS");

while(strcmp(input,"E")!=0)

{

if(strcmp(input,"H")==0)

{

fscanf(fp1,"%s %x %x %s",label,&start,&end,input);

address=start;

}

else if(strcmp(input,"T")==0)

{

l=length;

ptaddr=addr;

fscanf(fp1,"%x %x %s",&taddr,&length,input);

addr=taddr;
```

```
if(w==0)

{

ptaddr=address;

w=1;

}

for(k=0;k<(taddr-(ptaddr+l));k++)

{

address=address+1;

fprintf(fp2,"xx");

count++;

if(count==4)

{

fprintf(fp2,"  ");

i++;

if(i==4)

{

fprintf(fp2,"\n\n%x\t\t",address);

i=0;

}

count=0;

}

}

if(taddr==start)

fprintf(fp2,"\n\n%x\t\t",taddr);

fprintf(fp2,"%c%c",input[0],input[1]);

check();

fprintf(fp2,"%c%c",input[2],input[3]);

check();

fprintf(fp2,"%c%c",input[4],input[5]);
```

```
check();

fscanf(fp1,"%s",input);

}

else

{

fprintf(fp2,"%c%c",input[0],input[1]);

check();

fprintf(fp2,"%c%c",input[2],input[3]);

check();

fprintf(fp2,"%c%c",input[4],input[5]);

check();

fscanf(fp1,"%s",input);

}

}

/*fprintf(fp2,"\n——————————————————-\n");*/

fclose(fp1);

fclose(fp2);

printf("\n\n The contents of output file:\n\n");

fp2=fopen("OUTPUT.DAT","r");

ch2=fgetc(fp2);

while(ch2!=EOF)

{

printf("%c",ch2);

ch2=fgetc(fp2);

}

fclose(fp1);

fclose(fp2);

getch();

}
```

```
void check()
{
count++;
address++;
taddr=taddr+1;
if(count==4)
{
fprintf(fp2,"  ");
i++;
if(i==4)
{
fprintf(fp2,"\n\n%x\t\t",taddr);
i=0;
}
count=0;

}
}
```



```
                        ABSOLUTE LOADER

 The contents of output file:
MEMORY ADDRESS                  CONTENTS
1000            14103348  20390010  36281030  30101548
1010            20613C10  0300102A  0C103900  102D0C10
1020            36482061  0810334C  0000454F  46000003
1030            000000xx  xxxxxxxx  xxxxxxxx  xxxxxxxx
1040            xxxxxxxx  xxxxxx04  10300010  30E0205D
1050            30203FD8  205D2810  30302057  5490392C
1060            205E3820  3F
```

RELOCATION LOADER:

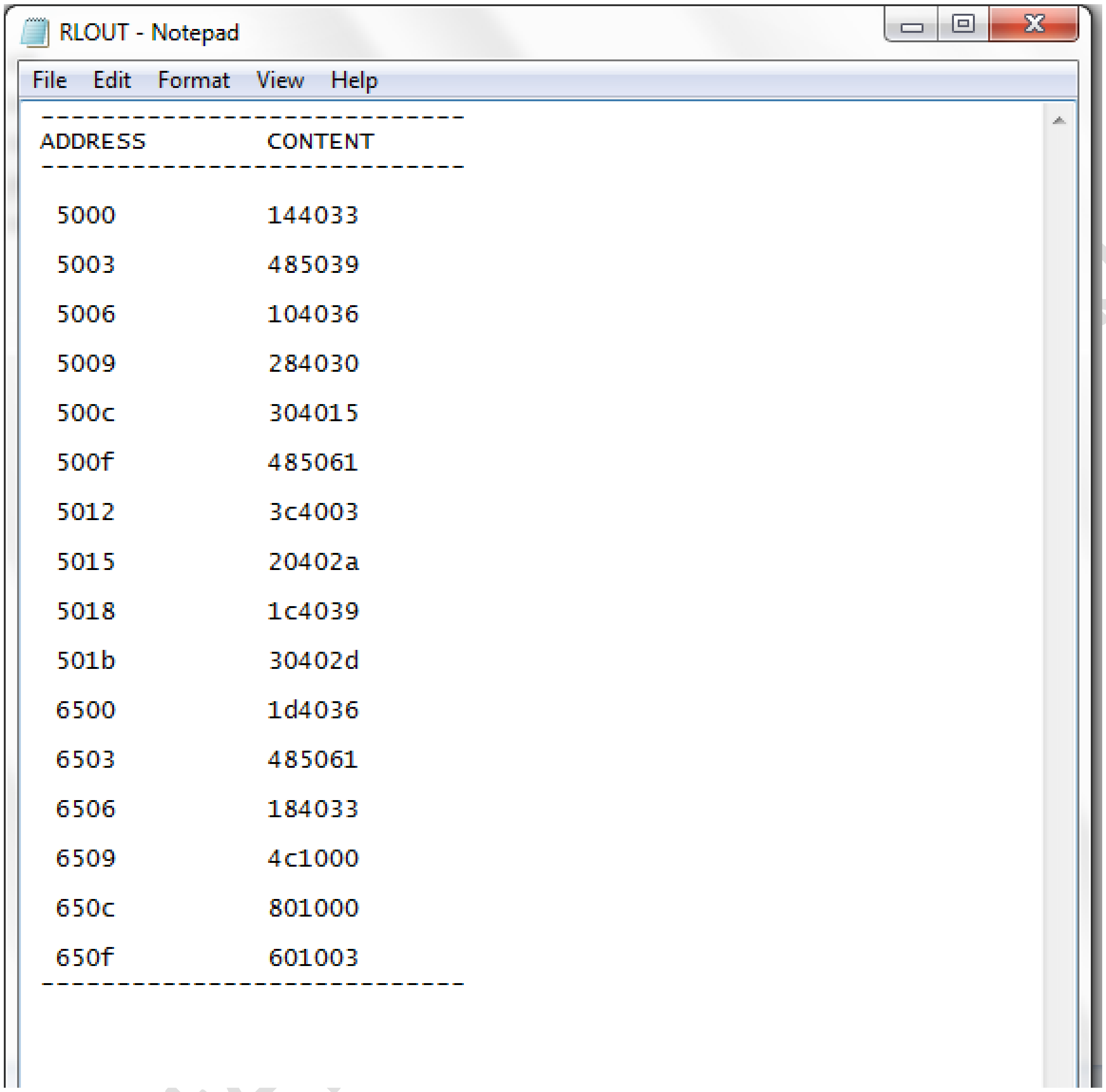
```
#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<stdlib.h>

void convert(char h[12]);

char bitmask[12];

char bit[12]={0};

int main()

{

        char add[6],length[10],input[10],binary[12],relocbit,ch,pn[5];

int start,inp,len,i,address,opcode,addr,actualadd,tlen;

FILE *fp1,*fp2;

//clrscr();

printf("\n\n Enter the actual starting address : ");

scanf("%x",&start);

fp1=fopen("RLIN.DAT","r");

fp2=fopen("RLOUT.DAT","w");

fscanf(fp1,"%s",input);

fprintf(fp2," -----------------------------\n");

fprintf(fp2," ADDRESS\tCONTENT\n");

fprintf(fp2," -----------------------------\n");

while(strcmp(input,"E")!=0)

{

if(strcmp(input,"H")==0)

{

fscanf(fp1,"%s",pn);

fscanf(fp1,"%x",add);

fscanf(fp1,"%x",length);
fscanf(fp1,"%s",input);
}
if(strcmp(input,"T")==0)
{
fscanf(fp1,"%x",&address);
fscanf(fp1,"%x",&tlen);
fscanf(fp1,"%s",bitmask);
address+=start;
convert(bitmask);
len=strlen(bit);
if(len>=11)
len=10;
for(i=0;i<len;i++)
{
fscanf(fp1,"%x",&opcode);
fscanf(fp1,"%x",&addr);
relocbit=bit[i];
if(relocbit=='0')
actualadd=addr;
else
actualadd=addr+start;
fprintf(fp2,"\n  %x\t\t%x%x\n",address,opcode,actualadd);
address+=3;
}
fscanf(fp1,"%s",input);
}
}
fprintf(fp2," ----------------------------\n");
int fcloseall(void);
```

```
printf("\n\n The contents of output file  is in RLOUT.DAT:\n\n");
fp2=fopen("RLOUT.DAT","r");
ch=fgetc(fp2);
while(ch!=EOF)
{
printf("%c",ch);
ch=fgetc(fp2);
}
fclose(fp2);
getch();
}
void convert(char h[12])
{
int i,l;
strcpy(bit,"");
l=strlen(h);
for(i=0;i<l;i++)
{
switch(h[i])
{
case '0':
   strcat(bit,"0");
   break;
case '1':
   strcat(bit,"1");
   break;
case '2':
   strcat(bit,"10");
   break;
```

```
case '3':
   strcat(bit,"11");
   break;
case '4':
   strcat(bit,"100");
   break;
case '5':
   strcat(bit,"101");
   break;
case '6':
   strcat(bit,"110");
   break;
case '7':
   strcat(bit,"111");
   break;
case '8':
   strcat(bit,"1000");
   break;
case '9':
   strcat(bit,"1001");
   break;
case 'A':
   strcat(bit,"1010");
   break;
case 'B':
   strcat(bit,"1011");
   break;
case 'C':
   strcat(bit,"1100");
   break;
case 'D':
   strcat(bit,"1101");
   break;
case 'E':
   strcat(bit,"1110");
   break;
case 'F':
   strcat(bit,"1111");
   break;
}
}
```

}

OUTPUT:



RLOUT - Notepad

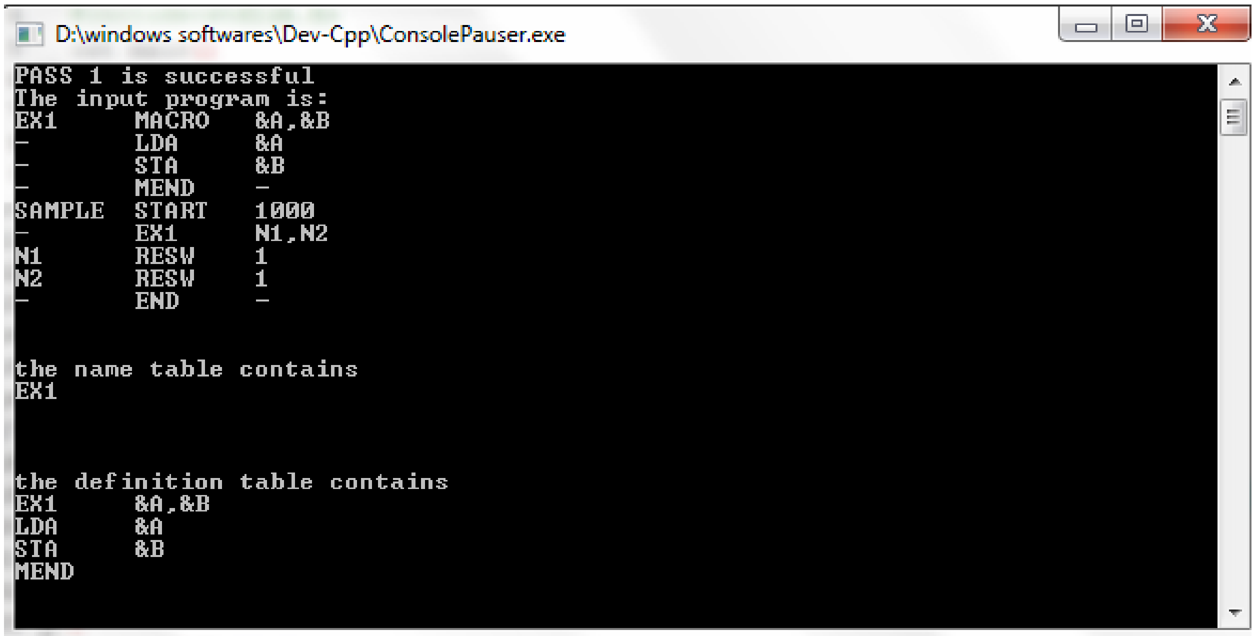
```
-----------------------------
ADDRESS          CONTENT
-----------------------------

 5000            144033

 5003            485039

 5006            104036

 5009            284030

 500c            304015

 500f            485061

 5012            3c4003

 5015            20402a

 5018            1c4039

 501b            30402d

 6500            1d4036

 6503            485061

 6506            184033

 6509            4c1000

 650c            801000

 650f            601003
-----------------------------
```

PASS1 MACROPROCESSOR:

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<stdlib.h>

int main()

{
```

```
FILE *f1,*f2,*f3;

char mne[20],opnd[20],la[20];

f1=fopen("minp2.txt","r");

f2=fopen("ntab2.txt","w+");

f3=fopen("dtab2.txt","w+");

fscanf(f1,"%s%s%s",la,mne,opnd);

while(strcmp(mne,"MEND")!=0)

{

if(strcmp(mne,"MACRO")==0)

{

fprintf(f2,"%s\n",la);

fprintf(f3,"%s\t%s\n",la,opnd);

}

else

fprintf(f3,"%s\t%s\n",mne,opnd);

fscanf(f1,"%s%s%s",la,mne,opnd);

}

fprintf(f3,"%s",mne);

fclose(f1);

fclose(f2);

fclose(f3);

printf("PASS 1 is successful\n");

FILE *fp1;

fp1=fopen("minp2.txt","r");

printf("The input program is:\n");

char ch;

 ch=fgetc(f1);

 while(ch!=EOF)

 {
    printf("%c",ch);

    ch=fgetc(f1);

  }

fclose(fp1);

printf("\n\n\nthe name table contains\n");

FILE *fp2;

fp2=fopen("ntab2.txt","r");

char ch1;

  ch1=fgetc(fp2);

  while(ch1!=EOF)

  {

    printf("%c",ch1);

    ch1=fgetc(fp2);

  }

fclose(fp2);

printf("\n\n\nthe definition table contains\n");

FILE *fp3;

fp3=fopen("dtab2.txt","r");

char ch2;

  ch2=fgetc(fp2);

  while(ch2!=EOF)

  {

    printf("%c",ch2);

    ch2=fgetc(fp2);

  }

fclose(fp3);

getch();

return 0;
```

}



```
PASS 1 is successful
The input program is:
EX1     MACRO   &A,&B
-       LDA     &A
-       STA     &B
-       MEND    -
SAMPLE  START   1000
-       EX1     N1,N2
N1      RESW    1
N2      RESW    1
-       END     -


the name table contains
EX1



the definition table contains
EX1     &A,&B
LDA     &A
STA     &B
MEND
```

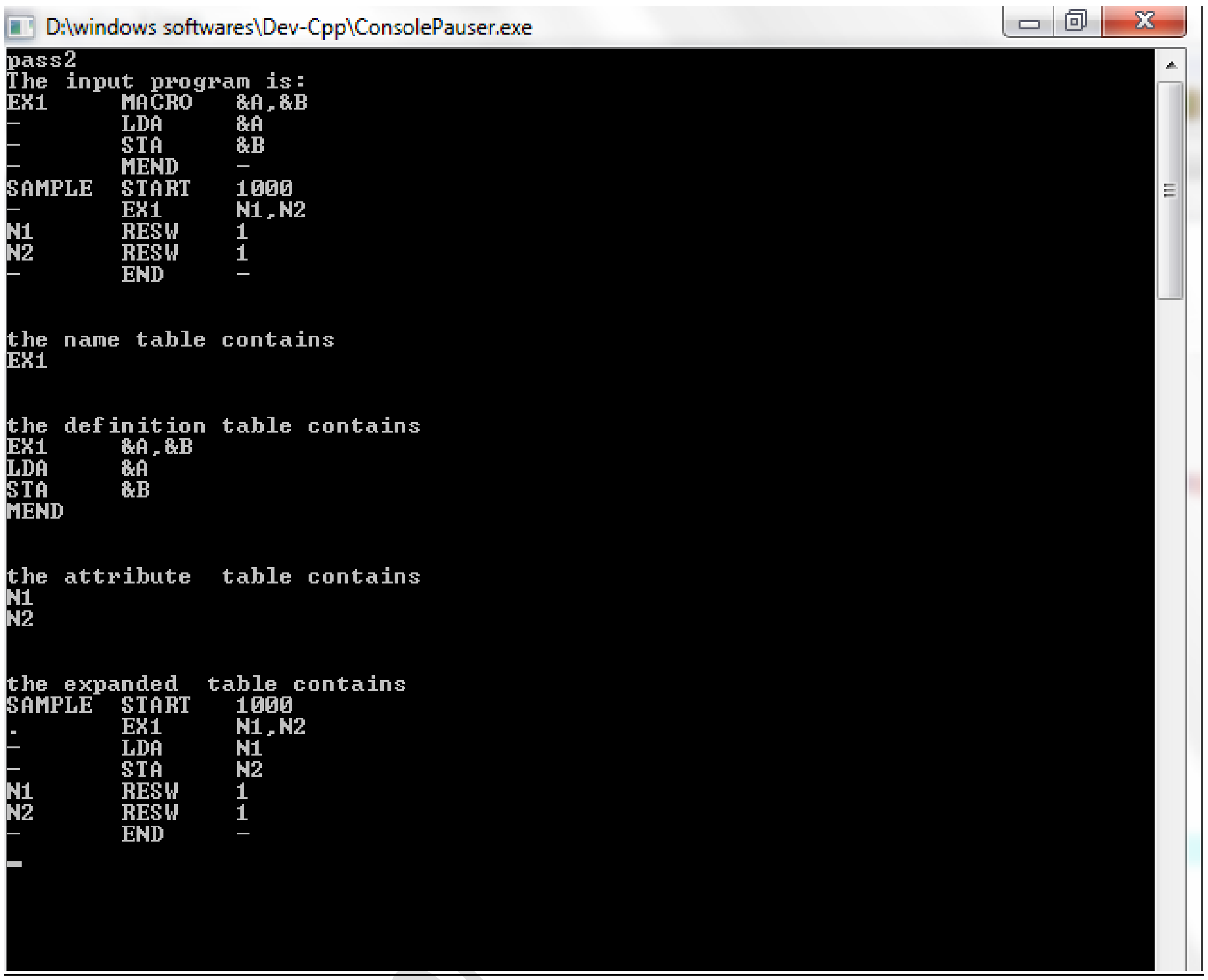
PASS2 MACROPROCESSOR:

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<stdlib.h>

int main()

{

FILE *f1,*f2,*f3,*f4,*f5;

int i,len;

char mne[20],opnd[20],la[20],name[20],mne1[20],opnd1[20],arg[20];

//clrscr();

f1=fopen("minp2.txt","r");

f2=fopen("ntab2.txt","r");

f3=fopen("dtab2.txt","r");

f4=fopen("atab2.txt","w+");

f5=fopen("op2.txt","w");

fscanf(f1,"%s%s%s",la,mne,opnd);

while(strcmp(mne,"END")!=0)
```

```
{
if(strcmp(mne,"MACRO")==0)
{
fscanf(f1,"%s%s%s",la,mne,opnd);
while(strcmp(mne,"MEND")!=0)
fscanf(f1,"%s%s%s",la,mne,opnd);
}
else
{
fscanf(f2,"%s",name);
if(strcmp(mne,name)==0)
{
len=strlen(opnd);
for(i=0;i<len;i++)
{
if(opnd[i]!=',')
fprintf(f4,"%c",opnd[i]);
else
fprintf(f4,"\n");
}
fseek(f2,SEEK_SET,0);
fseek(f4,SEEK_SET,0);
fscanf(f3,"%s%s",mne1,opnd1);
fprintf(f5,".\t%s\t%s\n",mne1,opnd);
fscanf(f3,"%s%s",mne1,opnd1);
while(strcmp(mne1,"MEND")!=0)
{
if((opnd1[0]=='&'))
{
```

```
fscanf(f4,"%s",arg);

fprintf(f5,"-\t%s\t%s\n",mne1,arg);

}

else

fprintf(f5,"-\t%s\t%s\n",mne1,opnd1);

fscanf(f3,"%s%s",mne1,opnd1);

}

}

else

fprintf(f5,"%s\t%s\t%s\n",la,mne,opnd);

}

fscanf(f1,"%s%s%s",la,mne,opnd);

}

fprintf(f5,"%s\t%s\t%s\n",la,mne,opnd);

fclose(f1);

fclose(f2);

fclose(f3);

fclose(f4);

fclose(f5);

printf("pass2\n");

FILE *fp1;

fp1=fopen("minp2.txt","r");

printf("The input program is:\n");

char ch;

 ch=fgetc(f1);

 while(ch!=EOF)

 {

   printf("%c",ch);

   ch=fgetc(f1);
```

```
  }

fclose(fp1);

printf("\n\n\nthe name table contains\n");

FILE *fp2;

fp2=fopen("ntab2.txt","r");

char ch1;

  ch1=fgetc(fp2);

  while(ch1!=EOF)

  {

     printf("%c",ch1);

     ch1=fgetc(fp2);

  }

fclose(fp2);

printf("\n\n\nthe definition table contains\n");

FILE *fp3;

fp3=fopen("dtab2.txt","r");

char ch2;

  ch2=fgetc(fp3);

  while(ch2!=EOF)

  {

     printf("%c",ch2);

     ch2=fgetc(fp2);

  }

fclose(fp3);

printf("\n\n\nthe attribute  table contains\n");

FILE *fp4;

fp4=fopen("atab2.txt","r");

char ch3;

  ch3=fgetc(fp4);
  while(ch3!=EOF)

 {

    printf("%c",ch3);

    ch3=fgetc(fp4);

 }

fclose(fp4);

printf("\n\n\nthe expanded  table contains\n");

FILE *fp5;

fp5=fopen("op2.txt","r");

char ch4;

  ch4=fgetc(fp5);

  while(ch4!=EOF)

 {

    printf("%c",ch4);

    ch4=fgetc(fp5);

 }

fclose(fp5);


getch();
}
```



```
pass2
The input program is:
EX1     MACRO   &A,&B
-       LDA     &A
-       STA     &B
-       MEND    -
SAMPLE  START   1000
-       EX1     N1,N2
N1      RESW    1
N2      RESW    1
-       END     -


the name table contains
EX1


the definition table contains
EX1     &A,&B
LDA     &A
STA     &B
MEND


the attribute  table contains
N1
N2


the expanded  table contains
SAMPLE  START   1000
.       EX1     N1,N2
-       LDA     N1
-       STA     N2
N1      RESW    1
N2      RESW    1
-       END     -
```

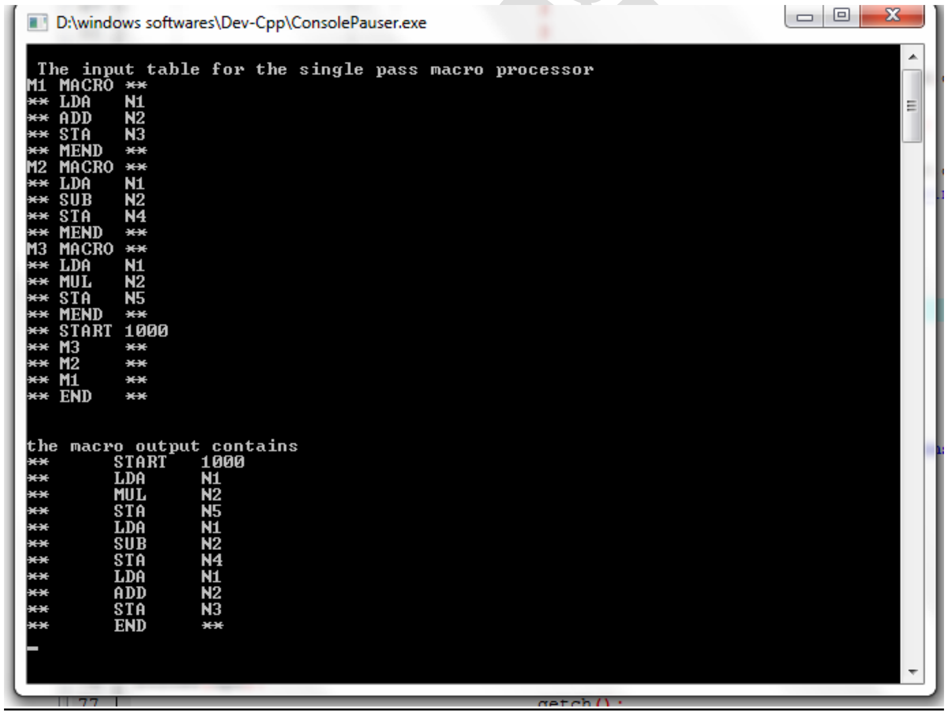
SINGLE PASS MACROPROCESSOR:

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<stdlib.h>

int main()

{

        int n,flag,i;

        char ilab[20],iopd[20],oper[20],NAMTAB[20][20];

        FILE *fp1,*fp2,*DEFTAB;



        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);
                                            printf("\n The input table for the
        single pass macro processor\n");
                                            FILE *f1;
                                            f1=fopen("MACROIN.DAT","r");
                                            char ch;
                                            ch=fgetc(f1);
                                            while(ch!=EOF)
                                            {
                                        printf("%c",ch);
                                        ch=fgetc(f1);
                                            }
                                      fclose(f1);
```

```
printf("\n\n\nthe macro output contains\n");

FILE *f2;

f2=fopen("macroout1.dat","r");

char ch1;

 ch1=fgetc(f2);

 while(ch1!=EOF)

 {

   printf("%c",ch1);

   ch1=fgetc(f2);

 }

fclose(fp2);

getch();

}
```

OUTPUT:

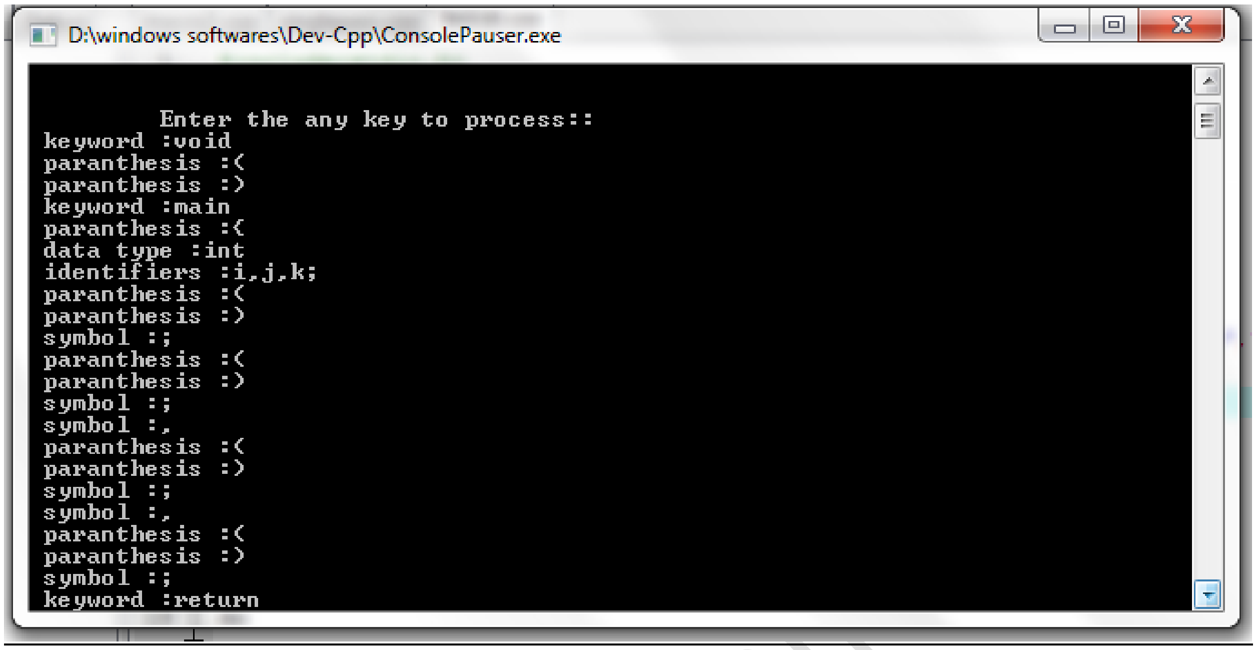


```
 The input table for the single pass macro processor
M1 MACRO **
** LDA   N1
** ADD   N2
** STA   N3
** MEND  **
M2 MACRO **
** LDA   N1
** SUB   N2
** STA   N4
** MEND  **
M3 MACRO **
** LDA   N1
** MUL   N2
** STA   N5
** MEND  **
** START 1000
** M3    **
** M2    **
** M1    **
** END   **


the macro output contains
**      START   1000
**      LDA     N1
**      MUL     N2
**      STA     N5
**      LDA     N1
**      SUB     N2
**      STA     N4
**      LDA     N1
**      ADD     N2
**      STA     N3
**      END     **
```

LEXICAL ANALYSER:

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

int main()

{

int i,j,k,p,c;

char s[120],r[100];

char par[6]={'(',')','{','}','[',']'};

char sym[9]={'.',';',':',',','<','>','?','$','#'};

char key[9][10]={"main","if","else","switch","void","do","while","for","return"};

char dat[4][10]={"int","float","char","double"};

char opr[5]={'*','+','-','/','^'};

FILE *fp;

printf("\n\n\t Enter the file name:: ");

scanf("%s",s);

fp=fopen(s,"r");

c=0;

printf("\n\n\t Enter the any key to process:: ");

do

{

fscanf(fp,"%s",r);

getch();

for(i=0;i<6;i++)

if(strchr(r,par[i])!=NULL)

printf("\n paranthesis :%c",par[i]);

for(i=0;i<9;i++)

if(strchr(r,sym[i])!=NULL)

printf("\n symbol :%c",sym[i]);
for(i=0;i<9;i++)
if(strstr(r,key[i])!=NULL)
printf("\n keyword :%s",key[i]);
for(i=0;i<4;i++)
 if((strstr(r,dat[i])&&(!strstr(r,"printf")))!=NULL)
{
printf("\n data type :%s",dat[i]);
fscanf(fp,"%s",r);
printf("\n identifiers :%s",r);
}
for(i=0;i<5;i++)
if(strchr(r,opr[i])!=NULL)
printf("\n operator :%c",opr[i]);
p=c;
c=ftell(fp);
}
while(p!=c);
return 0;
```

}



D:\windows softwares\Dev-Cpp\ConsolePauser.exe

```
        Enter the any key to process::
keyword :void
paranthesis :(
paranthesis :)
keyword :main
paranthesis :{
data type :int
identifiers :i,j,k;
paranthesis :(
paranthesis :)
symbol :;
paranthesis :(
paranthesis :)
symbol :;
symbol :,
paranthesis :(
paranthesis :)
symbol :;
symbol :,
paranthesis :(
paranthesis :)
symbol :;
keyword :return
```

<u>**parsetree:**</u>

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

char str[10],out,in,output[10],input[10],temp,ch;

char tl[10]={'x','+','*','(',')','$','@'};

char ntl[10]={'E','e','T','t','F'};

int err=0,flag=0,i,j,k,l,m;

char c[10][10][7]={{{"Te"},{"ERROR!"},{"ERROR!"},{"Te"},{"ERROR!"},{"ERROR!"}},

                        {"ERROR!","+Te","ERROR!","ERROR!","@","@"},

                        {"Ft","ERROR!","ERROR!","Ft","ERROR!","ERROR!"},

                        {"ERROR!","@","*Ft","ERROR!","@","@"},

                        {"x","ERROR!","ERROR!","(E)","ERROR!","ERROR!"}};


struct stack

        {

                char sic[10];
                int top;

        };


void push(struct stack *s,char p)

        {

                s->sic[++s->top]=p;

                s->sic[s->top+1]='\0';

        }


char pop(struct stack *s)

        {

                char a;

                a=s->sic[s->top];

                s->sic[s->top--]='\0';

                return(a);

        }


char sttop(struct stack *s)

        {

                return(s->sic[s->top]);

        }


void pobo(struct stack *s)

        {

        //printf("%s\n",str);

                m=0;

                while(str[m]!='\0')

                        m++;

                        m--;
                while(m!=-1)
                {
                        if(str[m]!='@')
                        push(s,str[m]);
                        m--;
                }
        }

void search(int l)
        {
                for(k=0;k<7;k++)
                        if(in==tl[k])
                                break;
                        if(l==0)
                        strcpy(str,c[l][k]);
                        else if(l==1)
                                strcpy(str,c[l][k]);
                                else if(l==2)
                                        strcpy(str,c[l][k]);
                                        else if(l==3)
                                                strcpy(str,c[l][k]);
                                                else strcpy(str,c[l][k]);
        }

int main()
{
        FILE *fp1,*fp2;
        struct stack s1;
        struct stack *s;
```

```
            s=&s1;

            s->top=-1;


            fp2=fopen("out.txt","w");

            fprintf(fp2,"\t\tPARSING TABLE\n~
      \t=============================================\n\n\tx\t+\t*\t(\t)\t$\n");

            fprintf(fp2," \t=============================================\n\n");

            for(i=0;i<5;i++)

            {

                  fprintf(fp2,"%c\t",ntl[i]);

                  for(j=0;j<6;j++)

                  if(strcmp(c[i][j],"ERROR!")==0)

                  fprintf(fp2,"ERROR!\t");

                  else

                  fprintf(fp2,"%c->%s\t",ntl[i],c[i][j]);

                  fprintf(fp2,"\n\n");

            }

            fprintf(fp2," \t=============================================\n\n");

      push(s,'$');

      push(s,'E');

      fp1=fopen("inp.txt","r");

      fscanf(fp1,"%s",input);


            fprintf(fp2,"\n\nTHE BEHAVIOUR OF THE PARSER FOR GIVEN INPUT STRING IS:\n\n");

            fprintf(fp2,"STACK\tINPUT\tOUTPUT\n");


                  i=0;

                  in=input[i];

            fprintf(fp2,"%s\t",s->sic);
```

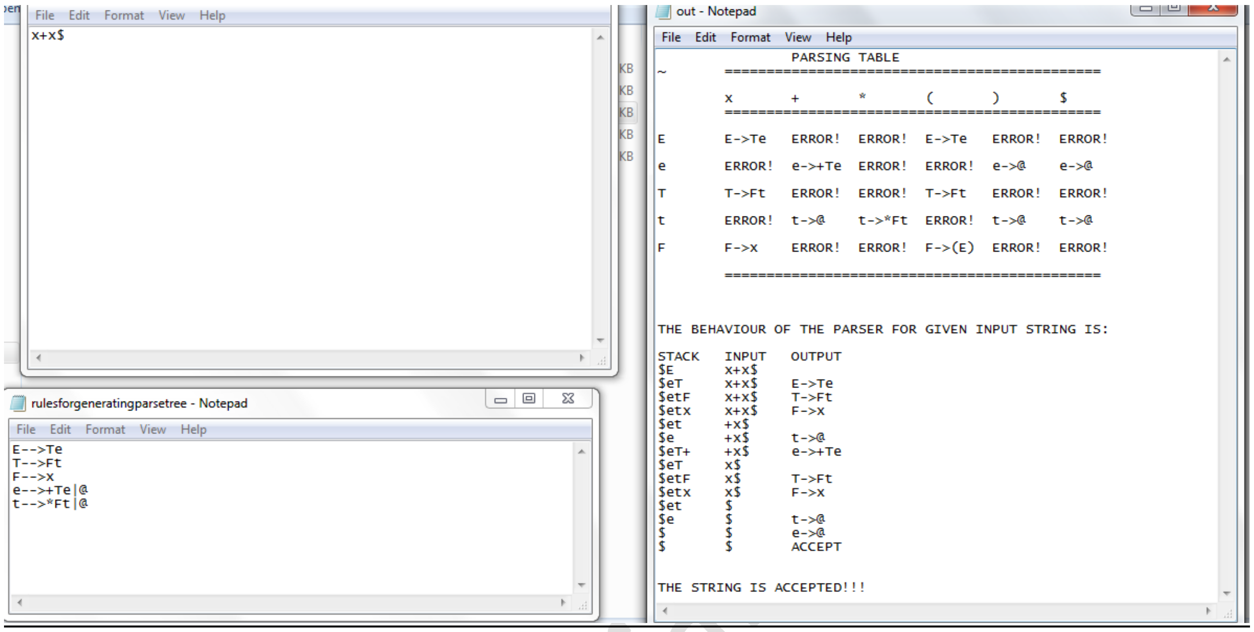
```
for(k=i;k<strlen(input);k++)

fprintf(fp2,"%c",input[k]);

if(strcmp(str,"")!=0)

fprintf(fp2,"\t%c->%s",ntl[j],str);


fprintf(fp2,"\n");

while((s->sic[s->top]!='$')&&err!=1&&strcmp(str,"ERROR!")!=0)

        {

                strcpy(str,"");

                flag=0;


                for(j=0;j<7;j++)

                if(in==tl[j])

                        {

                                flag=1;

                                break;

                        }

                if(flag==0)

                        in='x';



                flag=0;

                out=sttop(s);

                for(j=0;j<7;j++)

                if(out==tl[j])

                        {

                                flag=1;

                                break;
```

```
          }
if(flag==1)
{
          if(out==in)
          {
                    temp=pop(s);
                    in=input[++i];
              //if(str=='@')
              //temp=pop(s);
              //      err=1;
          }
          else
          {
                    strcpy(str,"ERROR!");
                    err=1;
          }
}
else
{
          flag=0;
          for(j=0;j<5;j++)
          if(out==ntl[j])
                    {
                              flag=1;
                              break;
                    }
          if(flag==1)
                    {
                              search(j);
```

```
                                                                        temp=pop(s);
                                                                        pobo(s);
                                                                }
                                                        else
                                                                {
                                                                        strcpy(str,"ERROR!");
                                                                        err=1;
                                                                }
                                                }
                                if(strcmp(str,"ERROR!")!=0)
                                {
                                fprintf(fp2,"%s\t",s->sic);
                                for(k=i;k<strlen(input);k++)
                                fprintf(fp2,"%c",input[k]);
                                if((strcmp(str,"")!=0)&&(strcmp(str,"ERROR!")!=0))
                                fprintf(fp2,"\t%c->%s",ntl[j],str);
                                fprintf(fp2,"\n");
                                }
                }
        if(strcmp(str,"ERROR!")==0)
        fprintf(fp2,"\n\nTHE STRING IS NOT ACCEPTED!!!!");

        else
        {
        fprintf(fp2,"$\t$\tACCEPT\n\n\nTHE STRING IS ACCEPTED!!!");
        //printf("\nTHE STRING IS ACCEPTED!!!");
        }
```

```
getch();

return 0;

}
```

output:



```
x+x$
```

rulesforgeneratingparsetree - Notepad

```
E-->Te
T-->Ft
F-->x
e-->+Te|@
t-->*Ft|@
```

out - Notepad

```
             PARSING TABLE
~       ============================================

        x       +       *       (       )       $
        ============================================

E       E->Te   ERROR!  ERROR!  E->Te   ERROR!  ERROR!

e       ERROR!  e->+Te  ERROR!  ERROR!  e->@    e->@

T       T->Ft   ERROR!  ERROR!  T->Ft   ERROR!  ERROR!

t       ERROR!  t->@    t->*Ft  ERROR!  t->@    t->@

F       F->x    ERROR!  ERROR!  F->(E)  ERROR!  ERROR!

        ============================================


THE BEHAVIOUR OF THE PARSER FOR GIVEN INPUT STRING IS:

STACK   INPUT   OUTPUT
$E      x+x$
$eT     x+x$    E->Te
$etF    x+x$    T->Ft
$etx    x+x$    F->x
$et     +x$
$e      +x$     t->@
$eT+    +x$     e->+Te
$eT     x$
$etF    x$      T->Ft
$etx    x$      F->x
$et     $
$e      $       t->@
$       $       e->@
$       $       ACCEPT

THE STRING IS ACCEPTED!!!
```