CS2308 - SYSTEM SOFTWARE LABORATORY

LABORATORY MANUAL

FOR FIFTH SEMESTER B.TECH-IT

ACADEMIC YEAR 2013-2014 (ODD)

(FOR PRIVATE CIRCULATION ONLY)

ANNA UNIVERSITY, CHENNAI

DEPARTMENT OF INFORMATION TECHNOLOGY

Dr. NAVALAR NEDUNCHEZHIYAN COLLEGE OF ENGINEERING

THOLUDUR 606303, CUDDALORE DIST.,

GENERAL INSTRUCTIONS FOR LABORATORY CLASSES

- Enter Lab with **CLOSED FOOTWEAR**
- Boys should **"TUCK IN"** the shirts
- Students should wear **uniform only**
- **LONG HAIR** should be protected, let it not be loose especially near **ROTATING MACHINERY.**
- Any other machines/ equipments **should not be operated** other than the prescribed one for that day.
- POWER SUPPLY to your test table should be obtained only through the **LAB TECHNICIAN**
- Do not **LEAN** and do not be **CLOSE** to the rotating components.
- **TOOLS, APPARATUS & GUAGE** Sets are to be returned before leaving the Lab.
- **HEADINGS & DETAILS** should be neatly written
    1. Aim of the experiment
    2. Apparatus / Tools/ Instruments required
    3. Procedure / Theory / Algorithm / Program
    4. Model Calculations
    5. Neat Diagram/ Flow charts
    6. Specifications/ Designs details
    7. Tabulation
    8. Graph
    9. Result/ Discussions

- Before doing the experiment, the student should get the circuit/ Program approval by the **FACULTY-IN-CHARGE**
- **Experiment date** should be written int the appropriate place
- After completing the experiments, the answer to the VIVA-VOCE Questions should be neatly written in the workbook
- Be **PATIENT, STEADY, SYSTEMATIC, & REGULAR**

HARDWARE REQUIREMENTS:

Processors	-	2.0 GHz or Higher
RAM	-	256 MB or Higher
Hard Disk	-	20 GB or Higher
Operating System	-	Windows 2000/XP/NT

SOFTWARE REQUIREMENTS:

TURBO C (Freeware)

UNIVERSITY PRACTICAL EXAMINATION
ALLOTMENT OF MARKS

Internal assessment	-	20 marks
Practical assessment	-	80 marks
Total	-	100 marks

INTERNAL ASSESSMENT (20 marks)

Staff should maintain the assessment Register and the Head of the Department should monitor it.

SPLIT UP OF INTERNAL MARKS

Record Note	-	10 marks
Model Exam	-	5 marks
Attendance	-	5 marks
Total	-	20 marks

UNIVERSITY EXAMINATION

The exam will be conducted for 100 marks. Then the marks will be calculated to 80 marks.

SPLIT UP OF PRACTICAL EXAMINATION MARKS

Aim and Algorithm	-	20 marks
Program	-	40 marks
Output	-	20 marks
Result	-	10 marks
Viva-voce	-	10 marks
Total	-	100 marks

CS2308 - SYSTEM SOFTWARE LAB

LIST OF EXPERIMENTS

1. Implement a symbol table with functions to create, insert, modify, search, and display.
2. Implement pass one of a two pass assembler.
3. Implement pass two of a two pass assembler.
4. Implement a single pass assembler.
5. Implement a two pass macro processor
6. Implement a single pass macro processor.
7. Implement an absolute loader.
8. Implement a relocating loader.
9. Implement pass one of a direct-linking loader.
10. Implement pass two of a direct-linking loader.
11. Implement a simple text editor with features like insertion / deletion of a character, word, and sentence.
12. Implement a symbol table with suitable hashing

CONTENTS

Exercise Number: 1

Title of the Exercise : IMPLEMENTATION OF A SYMBOL TABLE
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

- To write a program to implement symbol table.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities Required:

S.No.	Facilities required	Quantity
1	System	1
2	O/S	Windows XP
3	S/W name	C or C++

b) Procedure:

Step no	Details of the step
1	Initialize all the variables.
2	Design a menu through which we can create a symbol Table and perform operations as insert, modify, search and display.
3	Create a symbol table with fields as variable and value using create option. Entries may be added to the table while it is created itself.
4	Append new contents to the symbol table with the constraint that there is no duplication of entries, using insert option.
5	Modify existing content of the table using modify option.
6	Use display option to display the contents of the table.
7	End of the program.

c) Program:

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

Struct table
{
char var[10];   int value;
};
struct table tb1[20];
int i,j,n;          void create();             void modify();
int search(char variable[],int n);        void insert();              void display();
void main()
{         int ch,result=0;
char v[10];
clrscr();
do
{
printf("\n\n1.CREATE\n2.INSERT\n3.MODIFY\n4.SEARCH\n5.DISPLAY\n6.EXIT:\t");
scanf("%d",&ch);
```

```
switch(ch)
{
case 1:create();          break;
case 2:insert();          break;
case 3:modify();          break;

case 4:printf("\nEnter the variable to be searched:");
scanf("%s",&v);
result=search(v,n);
if(result==0)
printf("\nThe variable is not present\n");
else
printf("\nThe location of the variable is %d \n The value of %s is
%d.",result,tb1[result].var,tb1[result].value);
break;
case 5:display();         break;
case 6:exit(1);
}
}while(ch!=6);
getch();
}
void create()
{         printf("\nEnter the no. of entries:");
scanf("%d",&n);
printf("\nEnter the variable and the values:-\n");
for(i=1;i<=n;i++)
{ scanf("%s%d",tb1[i].var,&tb1[i].value);
check:
if(tb1[i].var[0]>='0' && tb1[i].var[0]<='9')
{ printf("\nVariable should start with alphabet\nEnter correct name\n");
scanf("%s%d",tb1[i].var,&tb1[i].value);
goto check;
}
check1:
for(j=1;j<i;j++)
{ if(strcmp(tb1[i].var,tb1[j].var)==0)
{ printf("\nThe variable already present. Enter another:");
scanf("%s%d",&tb1[i].var,&tb1[i].value);
goto check1;
}         }         }
printf("\nThe table after creation is:\n");
display();
}
void insert()
{
if(i>=20)
printf("\nCannot insert.table is full\n");
else
{
n++;
```

```
printf("\nEnter the variable and the value:");
scanf("%s%d",&tb1[n].var,&tb1[n].value);
check:
if(tb1[i].var[0]>='0' && tb1[i].var[0]<='9')
{
printf("\nVariable should start with alphabet\nEnter correct name\n");
scanf("%s%d",tb1[i].var,&tb1[i].value);
goto check;
}
check1:
for(j=1;j<n;j++)
{
if(strcmp(tb1[j].var,tb1[i].var)==0)
{
printf("\nThe variable already present. Enter another:");
scanf("%s%d",&tb1[i].var,&tb1[i].value);
goto check1;
}
}
printf("\nThe table after insertion is:");
display();
}
}
void modify()
{
char variable[10];
int result=0;
printf("\nEnter the variable to be modified:");
scanf("%s",&variable);
result=search(variable,n);
if(result==0)
printf("%s not present\n",variable);
else
{
printf("\nThe current value of the variable %s is %d.\nEnter the new variable and its
value",tb1[result].var,tb1[result].value);
scanf("%s%d",tb1[result].var,&tb1[result].value);
check:
if(tb1[i].var[0]>='0' && tb1[i].var[0]<='9')
{
printf("\nVariable should start with alphabet\nEnter correct name\n");
scanf("%s%d",tb1[i].var,&tb1[i].value);
goto check;
}
}
printf("\nThe table after modification is:");
display();
}

int search(char variable[],int n)
```

```
{
int flag;
for(i=1;i<=n;i++)
if(strcmp(tb1[i].var,variable)==0)
{
flag=1;
break;
}
if(flag==1)
return i;
else
return 0;
}

void display()
{
printf("\nVariable\tvalue\n");
for(i=1;i<=n;i++)
printf("%s\t\t%d\n",tb1[i].var,tb1[i].value);
}
```

d) Output:

```
1.CREATE
2.INSERT
3.MODIFY
4.SEARCH
5.DISPLAY
6.EXIT:1

Enter the no. of entries:2
Enter the variable and the values:-
a 23
c 45

The table after creation is:
Variable      value
a             23
c             45
1.CREATE 2.INSERT 3.MODIFY 4.SEARCH 5.DISPLAY 6.EXIT:2

Enter the variable and the value:b 34
The table after insertion is:
Variable      value
a             23
c             45
b             34
1.CREATE 2.INSERT 3.MODIFY 4.SEARCH 5.DISPLAY 6.EXIT: 3

Enter the variable to be modified:c
```

```
The current value of the variable c is 45.
Enter the new variable and its valuec 44
The table after modification is:
Variable       value
a              23
c              44
b              34

1.CREATE 2.INSERT 3.MODIFY 4.SEARCH 5.DISPLAY 6.EXIT: 4

Enter the variable to be searched:a
The location of the variable is 1
 The value of a is 23.

1.CREATE 2.INSERT 3.MODIFY 4.SEARCH 5.DISPLAY 6.EXIT: 5
Variable       value
a              23
c              44
b              34

1.CREATE 2.INSERT 3.MODIFY 4.SEARCH 5.DISPLAY 6.EXIT: 6
```

e) Result:

Thus the symbol table is created and operations are verified successfully.

VIVA – QUESTION AND ANSWER:

1. What is system software?

System software consists of variety of programs that supports the operations of a computer. This makes it possible for the user to focus on an application or other problem to be solved, without needing to know the details of how the machine works internally.
Examples of system software are text-editors, compilers, loaders or linkers, debuggers, assemblers, and operating systems.

2. Give some applications of operating system.

- to make the computer easier to use
- to manage the resources in computer
- process management
- data and memory management
- to provide security to the user.
- Operating system acts as an interface between the user and the system
  Eg: windows, Linux, UNIX, dos

3. What is SIC machine?

SIC refers to Simplified Instruction Computer which is a hypothetical computer that has been designed to include the hardware features most often found on real machines, while avoiding unusual and irrelevant complexities. This allows to clearly separating the central concepts of system software from the implementation details associated with a particular machine

Exercise Number: 2

Title of the Exercise : IMPLEMENTATION OF PASS ONE OF A TWO PASS ASSEMBLER

Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To write a program to implement pass one of a two pass assembler.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities Required:

S.No.	Facilities required	Quantity
1	System	1
2	O/S	Windows XP
3	S/W name	C or C++

b) Procedure:

Step no.	Details of the step
1	begin
2	read first input line;
3	if OPCODE = 'START' then begin i. save #[OPERAND] as starting address ii. initialized LOCCTR to starting address iii. write line to intermediate file iv. read next input line end {if START}
4	else initialized LOCCTR to 0
5	while OPCODE != 'END' do begin a. i. if this is not a comment line then ii. begin iii. if there is a symbol in the LABEL field then begin 1. search SYMTAB for LABEL 2. if found then 3. set error flag (duplicate symbol) 4. else 5. insert (LABEL, LOCCTR) into SYMTAB end {if symbol} iv. search OPTAB for OPCODE v. if found then add 3 {instruction length} to LOCCTR vi. else if OPCODE = 'WORD' then add 3 to LOCCTR vii. else if OPCODE = 'RESW' then

	add 3 * #[OPERAND] to LOCCTR viii. else if OPCODE = 'RESB' then add #[OPERAND] to LOCCTR ix. else if OPCODE = 'BYTE' then begin 1. find length of constant in bytes 2. add length to LOCCTR end {if BYTE} x. else set error flag (invalid operation code) end {if not a comment}
6	write line to intermediate file
7	read next input line
8	end {while not END}
9	write last line to intermediate file
10	save (LOCCTR - starting address) as program length
11	end

c) Program:

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

void main()
{
char opcode[10],mnemonic[3],operand[10],label[10],code[10];
int locctr,start,length;
FILE *fp1,*fp2,*fp3,*fp4;
clrscr();
fp1=fopen("input.txt","r");
fp2=fopen("symtbl.txt","w");
fp3=fopen("out.txt","w");
fp4=fopen("optab.txt","r");
fscanf(fp1,"%s%s%s",label,opcode,operand);
if(strcmp(opcode,"START")==0)
{
start=atoi(operand);
locctr=start;
fprintf(fp3,"\t%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);
fscanf(fp4,"%s%s",code,mnemonic);
while(strcmp(code,"END")!=0)
{
if(strcmp(opcode,code)==0)
{
locctr+=3;
break;
}
fscanf(fp4,"%s%s",code,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\t\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("The length of the program is %d",length);
fclose(fp1);
fclose(fp2);
fclose(fp3);
fclose(fp4);
getch();
}
```

d) Output:

<u>INPUT FILES:</u>

<u>input.txt</u>

```
**      START         2000
**      LDA    FIVE
**      STA    ALPHA
**      LDCH CHARZ
**      STCH  C1
ALPHA           RESW 1
FIVE   WORD          5
CHARZ           BYTE  C'Z'
C1      RESB   1
**      END    **
```

optab.txt

```
START       *
LDA   03
STA   0f
LDCH 53
STCH  57
END   *
```

OUTPUT FILES:

The length of the program is 20

symtab.txt

```
ALPHA       2012
FIVE  2015
CHARZ       2018
C1    2019
```

ouput.txt

```
      **    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'Z'
2019  C1    RESB  1
2020  **    END   **
```

e) Result:

Thus pass one of two passes assembler is implemented and the result is verified successfully.

VIVA – QUESTION AND ANSWER:

1. What are the instruction set for SIC/XE?

Instruction Set

a. New registers: LDB, STB, etc.
b. floating-point arithmetic: ADDF, SUBF, MULF, DIVF
c. register move: RMO
d. register-register arithmetic: ADDR, SUBR, MULR, DIVR
e. supervisor call: SVC- generates an interrupt for OS (Chap 6)

Input/Output

f. SIO, TIO, HIO: start, test, halt the operation of I/O device

2. Define loader.

Loader is a set of program that loads the machine language translated by the translator into the main memory and makes it ready for execution.

Exercise Number: 3

Title of the Exercise : IMPLEMENTATION OF PASS TWO OF A TWO PASS ASSEMBLER

Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To write program to implement pass two of a two pass assembler.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities Required:

S.No.	Facilities required	Quantity
1	System	1
2	O/S	Windows XP
3	S/W name	C or C++

b) Procedure:

Step no.	Details of the step
1	begin
2	read first input file {from intermediate file}
3	if OPCODE = 'START' then
4	begin a) write listing line b) read next input line
5	end {if START}
6	write header record to object program
7	initialized first Text record
8	while OPCODE != 'END' do
9	begin a) if this is not a comment line then i) begin ii) search OPTAB for OPCODE iii) if found then iv) begin (1) if there is a symbol in OPERAND field then (2) begin (a) search SYMTAB for OPERAND (b) if found then (c) store symbol value as operand address (d) else (e) begin (f) store 0 as operand address (g) set error flag (undefined symbol)

	(h) end (3) end {if symbol} (4) else (5) store 0 as operand address (6) assemble the object code instruction i) end {if opcode found} ii) else if OPCODE = 'BYTE' or 'WORD' then iii) convert constant to object code iv) if object code not fit into the current Text record then v) begin (7) write Text record to object program (8) initialized new Text record vi) end vii) add object code to Text record viii) end {if not comment} b) write listing line read next input line
10	end {while not END}
11	write last Text record to object program
12	write End record to object program
13	write last listing line
14	end

c) Program:

```
#include<stdio.h>            #include<conio.h>               #include<string.h>
void main()
{
char
opcode[10],operand[10],symbol[10],label[10],code[10],mnemonic[5],character,add[10],objectco
de[10];
int flag,flag1,locctr,location,loc;
FILE *fp1,*fp2,*fp3,*fp4;
clrscr();
fp1=fopen("out.txt","r");             fp2=fopen("twoout.txt","w");
fp3=fopen("optab.txt","r");           fp4=fopen("symtbl.txt","r");
fscanf(fp1,"%s%s%s",label,opcode,operand);
if(strcmp(opcode,"START")==0)
{   fprintf(fp2,"%s\t%s\t%s\n",label,opcode,operand);
    fscanf(fp1,"%d%s%s%s",&locctr,label,opcode,operand);
}
while(strcmp(opcode,"END")!=0)
{   flag=0;
    fscanf(fp3,"%s%s",code,mnemonic);
while(strcmp(code,"END")!=0)
{   if((strcmp(opcode,code)==0) && (strcmp(mnemonic,"*"))!=0)
    {        flag=1;
             break;
    }
fscanf(fp3,"%s%s",code,mnemonic);
}
if(flag==1)
{   flag1=0;        rewind(fp4);
while(!feof(fp4))
{
fscanf(fp4,"%s%d",symbol,&loc);
if(strcmp(symbol,operand)==0)
{
flag1=1;            break;
}           }
if(flag1==1)
{
itoa(loc,add,10);
strcpy(objectcode,strcat(mnemonic,add));
}           }
else if(strcmp(opcode,"BYTE")==0 || strcmp(opcode,"WORD")==0)
{
if((operand[0]=='C') || (operand[0]=='X'))
{
character=operand[2];
itoa(character,add,16);
strcpy(objectcode,add);
}
else
{
itoa(atoi(operand),add,10);
strcpy(objectcode,add);
}           }
else
strcpy(objectcode,"\0");
fprintf(fp2,"%s\t%s\t%s\t%d\t%s\n",label,opcode,operand,locctr,objectcode);
fscanf(fp1,"%d%s%s%s",&locctr,label,opcode,operand);
}
fprintf(fp2,"%s\t%s\t%s\t%d\n",label,opcode,operand,locctr);
fclose(fp1);                fclose(fp2);
fclose(fp3);                fclose(fp4);
getch();
}
```

d) Output:

INPUT FILES:
out.txt

```
      **        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'Z'
2019   C1           RESB         1
2020   **           END          **
```

optab.txt

```
START        *
LDA          03
STA          0f
LDCH         53
STCH         57
END          *
```

symtbl.txt

```
ALPHA        2012
FIVE         2015
CHARZ        2018
C1           2019
```

OUTPUT FILES:

twoout.txt

```
**           START                     2000
**           LDA          FIVE         2000         032015
**           STA          ALPHA        2003         0f2012
**           LDCH         CHARZ        2006         532018
**           STCH         C1           2009         572019
ALPHA        RESW         1            2012
FIVE         WORD               5              2015         5
CHARZ        BYTE         C'Z'         2018         5a
C1           RESB         1            2019
**           END          **           2020
```

e) Result:

Thus pass two of two pass assembler is implemented and the result is verified successfully.

VIVA – QUESTION AND ANSWER:

1. Define the basic functions of assembler.

* Translating mnemonic operation codes to their machine language equivalents.
* Assigning machine addresses to symbolic labels used by the programmer.

2. What is meant by assembler directives? Give example.

These are the statements that are not translated into machine instructions, but they Provide instructions to assembler itself.

Example START, END, BYTE, WORD, RESW and RESB

3. What is the need of SYMTAB (symbol table) in assembler?

The symbol table includes the name and value for each symbol in the source program, together with flags to indicate error conditions. Some times it may contain details about the data area. SYMTAB is usually organized as a hash table for efficiency of insertion and retrieval.

Exercise Number: 4

Title of the Exercise : IMPLEMENTATION OF A SINGLE PASS ASSEMBLER
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To write a program to implement a single pass assembler.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities Required:

S.No.	Facilities required	Quantity
1	System	1
2	O/S	Windows XP
3	S/W name	C or C++

b) Procedure:

Step no	Details of the step
1	Begin
2	Read first input line
3	if OPCODE='START' then a. save #[operand] as starting address b. initialize LOCCTr as starting address c. read next input line end
4	else initialize LOCCTR to 0
5	while OPCODE != 'END' do d. if there is not a comment line then e. if there is a symbol in the LABEL field then i. search SYMTAB for LABEL ii. if found then 1. if symbol value as null 2. set symbol value as LOCCTR and search the linked list with the corresponding operand 3. PTR addresses and generate operand addresses as corresponding symbol values 4. set symbol value as LOCCTR in symbol table and delete the linked list iii. end iv. else insert (LABEL,LOCCTR) into SYMTAB v. end
6	search OPTAB for OPCODE
7	if found then search SYMTAB for OPERAND address
8	if found then

	f. if symbol value not equal to null then i) store symbol value as operand address else insert at the end of the linked list with a node with address as LOCCTR
9	else insert (symbol name, null) add 3 to LOCCTR.
10	elseif OPCODE='WORD' then add 3 to LOCCTR & convert comment to object code
11	elseif OPCODE = 'RESW' then add 3 #[OPERND] to LOCCTR
12	elseif OPCODE = 'RESB' then add #[OPERND] to LOCCTR
13	elseif OPCODE = 'BYTE' then g. find length of the constant in bytes h. add length to LOCCTR convert constant to object code
14	if object code will not fit into current text record then i. write text record to object program j. initialize new text record o. add object code to text record
15	write listing line
16	read next input line
17	write last text record to object program
18	write end record to object program
19	write last listing line
20	End

c) Program:

```
#include<stdio.h>            #include<conio.h>
#include<string.h>           #include<stdlib.h>              #define MAX 10
struct input
{
char label[10],opcode[10],operand[10],mnemonic[5];                        int loc;
};
struct input table[MAX];
struct symtab
{
char sym[10];                int f,val,ref;
};
struct symtab symtbl[MAX];

void main()
{
int f1,i=1,j=1,flag,locctr,x;
char add[10],code[10],mnemcode[5];
FILE *fp1,*fp2,*fp3;
clrscr();
fp1=fopen("input1.txt","r");         fp2=fopen("optab1.txt","r");   fp3=fopen("spout.txt","w");
fscanf(fp1,"%s%s%s",table[i].label,table[i].opcode,table[i].operand);
if(strcmp(table[i].opcode,"START")==0)
{
locctr=atoi(table[i].operand);
```

```
i++;
fscanf(fp1,"%s%s%s",table[i].label,table[i].opcode,table[i].operand);
}
else
locctr=0;
while(strcmp(table[i].opcode,"END")!=0)
{    if(strcmp(table[i].label,"**")!=0)
     {
for(x=1;x<j;x++)
{  f1=0;
if((strcmp(symtbl[x].sym,table[i].label)==0) && (symtbl[x].f==1))
{
symtbl[x].val=locctr;        symtbl[x].f=0;
table[symtbl[x].ref].loc=locctr;                f1=1;
break;
}
}
if(f1==0)
{
strcpy(symtbl[j].sym,table[i].label);        symtbl[j].val=locctr;   symtbl[j].f=0;
j++;
}
}
fscanf(fp2,"%s%s",code,mnemcode);
while(strcmp(code,"END")!=0)
{    if(strcmp(table[i].opcode,code)==0)
     { strcpy(table[i].mnemonic,mnemcode);
      locctr+=3;
for(x=1;x<=j;x++)
{
flag=0;
if(strcmp(table[i].operand,symtbl[x].sym)==0)
{
flag=1;
if(symtbl[x].f==0)
table[i].loc=symtbl[x].val;
break;
}
}
if(flag!=1)
{
strcpy(symtbl[j].sym,table[i].operand);
symtbl[j].f=1;
symtbl[j].ref=i;
j++;
}
}
fscanf(fp2,"%s%s",code,mnemcode);
}
rewind(fp2);
```

```
if(strcmp(table[i].opcode,"WORD")==0)
{
locctr+=3;
strcpy(table[i].mnemonic,'\0');
table[i].loc=atoi(table[i].operand);
}
else if(strcmp(table[i].opcode,"RESW")==0)
{
locctr+=(3*(atoi(table[i].operand)));
strcpy(table[i].mnemonic,'\0');
table[i].loc=atoi('\0');
}
else if(strcmp(table[i].opcode,"RESB")==0)
{
locctr+=(atoi(table[i].operand));
strcpy(table[i].mnemonic,'\0');
table[i].loc=atoi('\0');
}
else if(strcmp(table[i].opcode,"BYTE")==0)
{
++locctr;
if((table[i].operand[0]=='C') || (table[i].operand[0]=='X'))
table[i].loc=(int)table[i].operand[2];
else
table[i].loc=locctr;
}
i++;
fscanf(fp1,"%s%s%s",table[i].label,table[i].opcode,table[i].operand);
}
for(x=1;x<=i;x++)
fprintf(fp3,"%s\t%s\t%s\t%s\n",table[x].label,table[x].opcode,table[x].operand,strcat(table[x].mn
emonic,itoa(table[x].loc,add,10)));
for(x=1;x<j;x++)
printf("%s\t%d\n",symtbl[x].sym,symtbl[x].val);
getch();
}
```

d) Output:

INPUT FILES:

input1.txt

```
**          START       6000
**          JSUB        CLOOP
**          JSUB        RLOOP
ALPHA       WORD        23
BETA        RESW        3
GAMMA       BYTE        C'Z'
DELTA       RESB        4
CLOOP       LDA         ALPHA
RLOOP       STA         BETA
**          LDCH        GAMMA
**          STCH        DELTA
**          END         **
```

optab1.txt

```
START       *
JSUB        48
LDA         14
STA         03
LDCH        53
STCH        57
END         *
```

OUTPUT FILES:

```
CLOOP   6023
RLOOP   6026
ALPHA   6006
BETA    6009
GAMMA   6018
DELTA   6019
```

spout.txt

```
**          START       6000        0
**          JSUB        CLOOP       486023
**          JSUB        RLOOP       486026
ALPHA       WORD            23          23
BETA        RESW        3           0
GAMMA       BYTE        C'Z'        90
DELTA       RESB        4           0
CLOOP       LDA         ALPHA       146006
RLOOP       STA         BETA        036009
**          LDCH        GAMMA       536018
**          STCH        DELTA       576019
**          END         **          0
```

e) Result:

Thus the program for single pass assembler is implemented and the output is verified accordingly.

VIVA – QUESTION AND ANSWER:

1. Define load and go assembler.

One pass assembler that generates their object code in memory for immediate execution is known as load and go assembler. Here no object programmer is written out and hence no need for loader.

2. What are the two different types of jump statements used in MASM assembler?

- . Near jump: A near jump is a jump to a target in the same segment and it is assembled by using a current code segment CS.
- . Far jump: A far jump is a jump to a target in a different code segment and it is assembled by using different segment registers.

Exercise Number: 5

Title of the Exercise : IMPLEMENTATION OF A MACRO PROCESSOR
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To write a program to implement a macro processor.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities Required:

S.No.	Facilities required	Quantity
1	System	1
2	O/S	Windows XP
3	S/W name	C or C++

b) Procedure:

Step no.	Details of the step
1	Start the macro processor program.
2	Include the necessary header files and variable.
3	Open the three files a.f1=macin.dat with read privilege b. f2=macout.dat with write privilege c. f3= deftab.dat with write privilege
4	Get the variable form f1 file macin.dat for label,opcode,operand
5	Read the variable until the opcode is not is equal to zero
6	Then check if the opcode is equal to Macro if Macro Then Copy macroname=label a. Get the variable label, opcode, operand b. In these if condition perform the while loop until opcode is not equal to MEND c. Copy the variable d. close while loop and if condition e. else if opcode is equal to macro name Perform the for loop from 0 to length
7	Finally terminate the program.

c) Program:

```
#include<stdio.h>              #include<conio.h>
#include<stdlib.h>             #include<string.h>
void main()
{
char n1,n,c1,i;
char fn[10][10],ilab[20],iopd[20],m[20][3],oper[20],opd[20];
FILE *fp1,*fp2,*p[5];               clrscr();              n=0;
fp1=fopen("macin.txt","r");
while(!feof(fp1))
{  fscanf(fp1,"%s%s%s",ilab,iopd,oper);
if(strcmp(iopd,"macro")==0)
n++;
}
printf("No. of macros=%d\n",n);
n1=n;
printf("Enter the text filename\n");
for(i=0;i<n;i++)
{   scanf("%s",fn[i]);
p[i]=fopen(fn[i],"w");
}
n=0;
rewind(fp1);
while(!feof(fp1))
{   fscanf(fp1,"%s%s%s",ilab,iopd,oper);
if(strcmp(iopd,"macro")==0)
{   strcpy(m[n],oper);
fscanf(fp1,"%s%s%s",ilab,iopd,oper);
while(strcmp(iopd,"mend")!=0)
{  fprintf(p[n],"%s %s %s\n",ilab,iopd,oper);
fscanf(fp1,"%s%s%s",ilab,iopd,oper);
}
fclose(p[n]);
n++;
}
}
for(i=0;i<n1;i++)
p[i]=fopen(fn[i],"r");   fp2=fopen("outmac.txt","w");           rewind(fp1);
fscanf(fp1,"%s%s%s",ilab,iopd,oper);
while(!feof(fp1))
{ if(strcmp(iopd,"call")==0)
{ for(i=0;i<n1;i++)
{ if(strcmp(m[i],oper)==0)
{ rewind(p[i]);
fscanf(p[i],"%s%s%s",ilab,iopd,oper);
while(!feof(p[i]))
{ fprintf(fp2,"%s %s %s\n",ilab,iopd,oper);
c1=1;
fscanf(p[i],"%s%s%s",ilab,iopd,oper);
}
break;
}     }     }
if(c1!=1)
fprintf(fp2,"%s %s %s\n",ilab,iopd,oper);
c1=0;
fscanf(fp1,"%s%s%s",ilab,iopd,oper);
}
fprintf(fp2,"%s %s %s\n",ilab,iopd,oper);
}
```

d) Output:

INPUT FILE:
macin.txt

```
**      macro  m1
**      move          a,b
**      mend   ---
**      macro  m2
**      lda           b
**      mend   ---
**      start    1000
**      lda     a
**      call    m1
**      call    m2
**      add           a,b
```

OUPUT FILE:

```
No. of macros=2
Enter the text filename
outmac
macin
```

outmac.txt

```
**       macro m1
**       move  a,b
**      mend   ---
**      macro  m2
**      lda           b
**      mend   ---
**      start    1000
**      lda           a
**      move          a,b
**      lda           b
**      add           a,b
```

e) Result:

Thus the macro processor is implemented and the result is verified successfully.

VIVA – QUESTION AND ANSWER:

1. What are the basic functions of loaders?

- Loading – brings the object program into memory for execution
- Relocation – modifies the object program so that it can be loaded at an address different from the location originally specified
- Linking – combines two or more separate object programs and also supplies the information needed to reference them.

2. Define absolute loader.

The loader, which is used only for loading, is known as absolute loader.
e.g. Bootstrap loader

3. What is meant by bootstrap loader?

This is a special type of absolute loader which loads the first program to be run by the computer. (usually an operating system)

Exercise Number: 6

Title of the Exercise : IMPLEMENTATION OF AN ABSOLUTE LOADER
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To write a program to implement an absolute loader.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities Required:

S.No.	Facilities required	Quantity
1	System	1
2	O/S	Windows XP
3	S/W name	C or C++

b) Procedure:

Step no.	Details of the step
1	Begin
2	Read Header record
3	Verify program name and length
4	Read first Text record
5	while record type ≠ 'E' do
6	Begin a. {if object code is in character form, convert into i. internal representation} b. move object code to specified location in memory read next object program record
7	End
8	Jump to address specified in End record
9	End

c) Program:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
struct object_code
{
int locctr;
char byte[5];
};
struct object_code code[200];
```

```
void main()
{
FILE *fp1,*fp2;
char input[15];
int i,len,n=0,count=0,inc=0,textloc,tlen,tloc=0,num=0,loc;
clrscr();
fp1=fopen("loadin.txt","r");
fp2=fopen("loadout.txt","w");
rewind(fp1);
rewind(fp2);
fscanf(fp1,"%s",input);
if(strcmp(input,"H")==0)
{
for(i=0;i<4;i++)
{
if(i==1)
fscanf(fp1,"%x",&loc);
else
fscanf(fp1,"%s",input);
}
}
tloc=loc;
while(strcmp(input,"E")!=0)
{
if(strcmp(input,"T")==0)
{
fscanf(fp1,"%x",&textloc);
for(i=0;i<(textloc-(tloc+tlen));i++)
{
strcpy(code[inc].byte,"xx");
code[inc++].locctr=loc++;
}
fscanf(fp1,"%x",&tlen);
tloc=textloc;
}
else
{
len=strlen(input);
for(i=0;i<len;i++)
{
code[inc].byte[num++]=input[i];
if(num > 1)
{
code[inc].locctr=loc;
loc++;
inc++;
num=0;
}
}
}
fscanf(fp1,"%s",input);
}
n=0;
i=0;
count=0;
fprintf(fp2,"%x\t",code[i].locctr);
for(i=0;i<inc;i++)
{
fprintf(fp2,"%s",code[i].byte);
n++;
if(n > 3)
{
fprintf(fp2,"\t");
n=0;
count++;
}
if(count > 3)
{
fprintf(fp2,"\n%x\t",code[i+1].locctr);
count=0;
}
}
getch();
}
```

d) Output:

INPUT FILE:

loadin.txt

```
H       COPY  002000        00107a
T       002000      1e      142033      483039      102036      282030      302015
        483061      3c200300202a0c203900202d
T       00201e15    2C2036      483062      182033      4C0000      454f46
        200003      100000
T   002039    1e    242030      302030      e0305d30303fd8305d
        282030      303057      53a0392c305e38303f
T       002057      a       102036      4c0000F1    201000
T       002071      19      342030      e03079303064      4fa039 dc3079
        2c2036383064      4c000015
E       002000
```

OUTPUT FILE:

loadout.txt

```
2000   14203348    30391020    36282030    30201548
2010   30613c20    0300202a    0c203900    202d2C20
2020   36483062    1820334C    0000454f    46200003
2030   100000xx    xxxxxxxx    xx242030    302030e0
2040   305d3030    3fd8305d    28203030    305753a0
2050   392c305e    38303f10    20364c00    00F12010
2060   00xxxxxx    xxxxxxxx    xxxxxxxx    xxxxxxxx
2070   xx342030    e0307930    30644fa0    39dc3079
       2080   2c203638    30644c00    0015
```

e) Result:

Thus the absolute loader is implemented and the result is verified successfully.

VIVA – QUESTION AND ANSWER:

1. Define bit mask.

The relocation bits are gathered together following the length indicator in each text record and which is called as bit mask. For e.g. the bit mask FFC (111111111100) specifies that the first 10 words of object code are to be modified during relocation.

2. What is the need of ESTAB?

It is used to store the name and address of the each external symbol. It also indicates in which control section the symbol is defined.

3. Write the two passes of a linking loader.

Pass1: assigns address to all external symbols
Pass2: it performs actual loading, relocation and linking.

4. Define automatic library search.

In many linking loaders the subroutines called by the program being loaded are automatically fetched from the library, linked with the main program and loaded. This feature is referred to as automatic library search.

5. Define dynamic linking.

If the subroutine is loaded and linked to the program during its first call (run time), then it is called as dynamic loading or dynamic linking.

6. Write the advantage of dynamic linking.

- It has the ability to load the routine only when they are needed.
- The dynamic linking avoids the loading of entire library for each execution.

Exercise Number: 7

Title of the Exercise : IMPLEMENTATION OF A RELOCATING LOADER
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To write a program to implement a relocating loader.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities Required:

S.No.	Facilities required	Quantity
1	System	1
2	O/S	Windows XP
3	S/W name	C or C++

b) Procedure:

Step no	Details of the step
1	Start the program.
2	Include the necessary header file and variable
3	Open the two file for a.fp1= relinput.dat and give read b.fp2= reloutput.dat and give write
4	Read the content using while loop perform the loop Until character is not equal to E **a.**while (strcmp (input,"E")!=0) i. If the character is H Get the variable add, length, and input ii.Else if the character is T Get the variable address and bitmask And perform the for loop for starting zero to up to len Get the opcode ,addr and assign relocbit to bitmask iii. If relocabit is zero Then actualadd=addr; iv**.** Else Add the addr and star value
5	Finally terminate the program.

c) Program:

```
#include<stdio.h>               #include<conio.h>     #include<string.h>     #include<stdlib.h>
struct object_code
{
int locctr;                 char add[10];             }obcode[300];
void main()
{ char input[100][16],output[100][16],binary[20],address[20],stloc[10];
int len,bitmask,loc,tlen=0,tloc,textloc,i=0,location,j,k,count=0,start,n,num=0,inc=0;
FILE *fp1,*fp2;                       clrscr();
fp1=fopen("relin.txt","r");           fp2=fopen("relout.txt","w");
```

```
printf("Enter the location where the present program has to be loaded:");
scanf("%s",stloc);       start=atoi(stloc);
location=start;
tloc=start;
fscanf(fp1,"%s",input[i]);
while(strcmp(input[i],"T")!=0)
{
strcpy(output[i],input[i]);
i++;
fscanf(fp1,"%s",input[i]);
strcpy(output[i],input[i]);
}
itoa(start,output[2],10);
while(strcmp(input[i],"E")!=0)
{
strcpy(output[i],input[i]);
if(strcmp(input[i],"T")==0)
{
for(j=0;j<3;j++)
{
i++;
fscanf(fp1,"%s",input[i]);
strcpy(output[i],input[i]);
}
bitmask=atoi(output[i]);
itoa(bitmask,binary,2);
strcpy(output[i],NULL);
textloc=atoi(output[i-2]);
textloc=textloc+start;
itoa(textloc,output[i-2],10);
for(n=0;n<(textloc-(tloc+tlen));n++)
{
strcpy(obcode[inc].add,"xx");
obcode[inc++].locctr=location++;
}
tlen=atoi(output[i-1]);
tloc=textloc;
k=0;
}
else
{
if(binary[k]=='1')
{
num=0;
len=strlen(output[i]);
strcpy(address,NULL);
for(j=2;j<len;j++)
{
address[num]=output[i][j];
output[i][j]='\0';
num++;
}
loc=atoi(address);                loc=loc+start;            itoa(loc,address,10);
strcat(output[i],address);
}
k++;
len=strlen(output[i]);            num=0;
for(n=0;n<len;n++)
{obcode[inc].add[num++]=output[i][n];
if(num>1)
{obcode[inc++].locctr=location++;
num=0;
}                   }                   }
i++;
fscanf(fp1,"%s",input[i]);
}
strcpy(output[i],input[i]);
i++;
fscanf(fp1,"%s",input[i]);
loc=atoi(input[i]);
loc=loc+start;
strcpy(output[i],itoa(loc,address,10));
count=0;
i=0;
n=0;
fprintf(fp2,"%d\t",obcode[n].locctr);
for(n=0;n<inc;n++)
{
fprintf(fp2,"%s",obcode[n].add);
i++;
if(i > 3)
{
fprintf(fp2,"\t");                i=0;               count++;
}
if(count > 3)
{
fprintf(fp2,"\n%d\t",obcode[n+1].locctr);               count=0;
}
}
getch();
}
```

d) Output:

INPUT FILE:

relin.txt

```
H COPY 000000 001073
T 000000 10 015 140033 481039 100036 280030 300015 481061 311003
200030 211033 200033
T 000011 19 045 412036 481061 380033 412000 454196 100003 200000
```

```
T 000031 15 135 140030 430030 141013 301044 241064 210030 301057
543039 212064 381045
T 000058 05 056 100036 520000 151 301000
T 000065 19 080 340030 141079 301064 503039 152079 220036 381064
430000 25
E 000000
```

OUTPUT FILE:

relout.txt

```
3000  14303348    40391030    36283030    30001548
3016  10613110    03200030    21103320    0033xx41
3032  50364810    61383033    41500045    41961030
3048  03200000    xx143030    43003014    10133010
3064  44241064    21303030    40575460    39212064
3080  381045xx    xxxxxxxx    xxxxxxxx    xxxxxx10
3096  30365230    00154000    301000xx    xx343030
3112  14107930    40645030    39152079    22003638
3128  10644300    0025
```

e) Result:

Thus the relocating loader is implemented and the result is verified successfully.

VIVA – QUESTION AND ANSWER:

1. Define macro processor.

Macro processor is system software that replaces each macroinstruction with the corresponding group of source language statements. This is also called as expanding of macros.

2. What do macro expansion statements mean?

These statements give the name of the macroinstruction being invoked and the arguments to be used in expanding the macros. These statements are also known as macro call.

3. What is the use of macro time variable?

Macro time variable can be used to store working values during the macro expansion. Any symbol that begins with the character & and then is not a macro instruction parameter is assumed to be a macro time variable.

4. What is meant by line by line processor?

This macro processor reads the source program statements, process the statements and then the output lines are passed to the language translators as they are generated, instead of being written in an expanded file.

5. Give the advantages of general-purpose macro processors.

- The programmer does not need to learn about a macro facility for each compiler.
- Overall saving in software development cost and maintenance cost.

6. What is the symbol used to generate unique labels?

$ Symbol is used in macro definition to generate unique symbols. Each macro expansion the $ symbol is replaced by $XX, where XX is the alpha numeric character.

Exercise Number: 8

Title of the Exercise : IMPLEMENTATION OF PASS 1 OF A DIRECT LINKING LOADER

Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

- To write programs to implement pass 1 of a direct linking loader.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities Required:

S.No.	Facilities required	Quantity
1	System	1
2	O/S	Windows XP
3	S/W name	C or C++

b) Procedure:

Step no.	Details of the step
1	Begin
2	get PROGADDR from operating system
3	set CSADDR to PROGADDR {for first control section}
4	while not end of input do
5	begin i. read next input record {Header record for control section} ii. set CSLTH to control section length iii. search ESTAB for control section name iv. if found then set error flag {duplicate external symbol} v. else enter control section name into ESTAB with value CSADDR
6	while record type ≠ 'E' do vi. begin 1. read next input record 2. if record type = 'D' then a. for each symbol in the record do b. begin i. search ESTAB for symbol name ii. if found then set error flag (duplicate external symbol) iii. else enter symbol into ESTAB with value (CSADDR + indicated address) end {for}

7	end {while ≠ 'E'}
8	add CSLTH to CSADDR {starting address for next control section
9	end {while not EOF}
10	end {Pass 1}

c) Program:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
#define MAX 10
struct estab
{
char csect[10];
char sym_name[10];
long int add;
int length;
}table[MAX];
void main()
{
FILE *fp1,*fp2;
char input[10];
long int i,count=0,start,length,loc;
clrscr();
fp1=fopen("dlinkin.txt","r");
fp2=fopen("dlinkout.txt","w");
printf("\nEnter the location where the program has to be loaded:");
scanf("%lx",&start);
fprintf(fp2,"CSECT\tSymname\tAddress\tLength\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,"\0");
fscanf(fp1,"%s",input);
table[count].add=atoi(input)+start;
fscanf(fp1,"%s",input);
length=atoi(input);
table[count++].length=atoi(input);
fscanf(fp1,"%s",input);
}
if(strcmp(input,"D")==0)
{
fscanf(fp1,"%s%lx",input,&loc);
while((strcmp(input,"R")!=0))
{
```

```
strcpy(table[count].csect,"\0");
strcpy(table[count].sym_name,input);
table[count].add=loc+start;
table[count++].length=0;
fscanf(fp1,"%s%lx",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%lx\t%d\n",table[i].csect,table[i].sym_name,table[i].add,table[i].length);
getch();
}
```

d) Output:

INPUT FILE:

dlinkin.txt

```
H PROGA      000000          000070
D LISTA 000040 ENDA 000054
R LISTB ENDB LISTC ENDC
T 000020 10 03201D 77100004 150014
T 000054 16 100014 15100006 00002F 100014 FFFFC0
M 000024 05 +LISTB
M 000054 06 +LISTC
M 000058 06 +ENDC
M 000064 06 +LISTB
E 000000
H PROGB 000000 000088
D LISTB 000060 ENDB 000070
R LISTA ENDA LISTC ENDC
T 000036 11 03100000 772027 05100000
T 000070 18 100000 05100006 05100020 05100030 100000
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 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
```

OUTPUT FILE:

dlinkout.txt

```
CSECT       Symname     Address     Length
PROGA                   2000        70
            LISTA 2040        0
            ENDA        2054        0
PROGB                   2046        88
            LISTB 20a6        0
            ENDB        20b6        0
PROGC                   209e        57
            LISTC 20ce        0
            ENDC        20e0        0
```

e) Result:

Thus the pass 1 of a direct linking loader is implemented and the result is verified successfully.

VIVA – QUESTION AND ANSWER:

1. Define interactive editor?

An interactive editor is a computer program that allows a user to create and revise a target document. The term document includes objects such as computer programs, text, equations, tables, diagrams, line art, and photographs any thing that one might find on a printed page.

2. What are the tasks performed in the editing process?

4 tasks

- Select the part of the target document to be viewed and manipulated.
- Determine how to format this view on-line and how to display it.
- Specify and execute operations that modify the target document.
- Update the view appropriately.

3. What are the three categories of editor's devices?

- Text device/ String devices
- Button device/Choice devices
- Locator device

4. What is the function performed in editing phase?

In the actual editing phase, the target document is created or altered with a set of operations such as insert, delete, replace, move and copy.

5. Define Locator device?

Locator devices are two-dimensional analog-to-digital converters that position a cursor symbol on the screen by observing the user's movement of the device. The most common such devices for editing applications are the mouse and the data tablet.

Exercise Number: 9

Title of the Exercise : IMPLEMENTATION OF PASS 2 OF A DIRECT LINKING LOADER

Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

- To write a program to implement pass 2 of a direct linking loader.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities Required:

S.No.	Facilities required	Quantity
1	System	1
2	O/S	Windows XP
3	S/W name	C or C++

b) Procedure:

Step no.	Details of the step
1	Begin
2	set CSADDR to PROGADDR
3	set EXECADDR to PROGADDR
4	while not end of input do
5	begin i. read next input record {Header record} ii. set CSLTH to control section length iii. while record type ≠'E' do iv. begin 1. read next input record 2. if record type = 'T' then a. begin b. {if object code is in character form, convert into internal representation} c. move object code from record to location i. (CSADDR + specified address) end {if 'T'}
6	else if record type = 'M' then d. begin e. search ESTAB for modifying symbol name f. if found then i. add or subtract symbol value at location ii. (CSADDR + specified address) g. else i. set error flag (undefined external symbol) h. end {if 'M'} v. end{while ≠'E'}

	vi. if an address is specified {in End record} then set EXECADDR to (CSADDR + specified address) add CSLTH to CSADDR .
7	end {while not EOF}
8	jump to location given by EXECADDR {to start execution of loaded program}
9	end {Pass 2}

c) Program:

```
#include<stdio.h>                #include<conio.h>
#include<string.h>               #include<stdlib.h>
struct ext_table
{char csect[10];                 char sname[10];                int padd;        int plen;
}estab[20];

struct object_code
{
char code[15];             int add;
}obcode[500];

void main()
{
FILE *fp1,*fp2,*fp3;
int i,j,x,y,n=0,num=0,inc=0,count=0,record=0,pstart,exeloc,start,textloc;
int loc,mloc[30],textlen,mlen[30],length,location;
long int newadd;
char *add1,operation,lbl[10],input[10],label[30][10],
address[10];
clrscr();
printf("\n this is pass 2 of direct linking loader\n");
fp1=fopen("dlink2in.txt","r");                  fp2=fopen("dlink1out.txt","r");
fp3=fopen("dlink2out.txt","w");                 rewind(fp1);
rewind(fp2);               rewind(fp3);
while(!feof(fp2))
{
fscanf(fp2,"%s%s%d%d",estab[num].csect,estab[num].sname,&estab[num].padd,&estab[num].plen);
num++;
}
exeloc=estab[0].padd;                  loc=exeloc;               start=loc;
while(!feof(fp1))
{
fscanf(fp1,"%s",input);
if(strcmp(input,"H")==0)
{
fscanf(fp1,"%s",input);
for(i=0;i<num;i++)
if(strcmp(input,estab[i].csect)==0)
{
pstart=estab[i].padd;                  break;
}
while(strcmp(input,"T")!=0)
fscanf(fp1,"%s",input);
}
do
{       if(strcmp(input,"T")==0)
        {       fscanf(fp1,"%d",&textloc);
        textloc=textloc+pstart;
for(i=0;i<(textloc-loc);i++)
{
strcpy(obcode[inc].code,"xx");
obcode[inc++].add=start++;
}
fscanf(fp1,"%d",&textlen);
loc=textloc+textlen;
}
else if(strcmp(input,"M")==0)
{
fscanf(fp1,"%d",&mloc[record]);              mloc[record]=mloc[record]+pstart;
fscanf(fp1,"%d",&mlen[record]);              fscanf(fp1,"%s",label[record++]);
}
else
{
length=strlen(input);
x=0;
for(i=0;i<length;i++)
{
obcode[inc].code[x++]=input[i];
if(x>1)
{
obcode[inc++].add=start++;
x=0;
}
}
}
fscanf(fp1,"%s",input);
}while(strcmp(input,"E")!=0);
if(strcmp(input,"E")==0)
fscanf(fp1,"%s",input);
}
for(n=0;n<record;n++)
{
operation=label[n][0];
length=strlen(label[n]);
for(i=1;i<length;i++)
lbl[i-1]=label[n][i];
lbl[length-1]='\0';
length=0;
strcpy(address,"\0");
location=mloc[n]-exeloc;
loc=location;
```

```
count=0;
while(length<mlen[n])
{
strcat(address,obcode[location++].code);
count++;
length+=2;
}
for(i=0;i<num;i++)
if(strcmp(lbl,estab[i].sname)==0)
break;
switch(operation)
{
case '+':newadd=strtol(address,&add1,10)+(long int)estab[i].padd;                    break;
case '-':newadd=strtol(address,&add1,10)-(long int)estab[i].padd;              break;
}
ltoa(newadd,address,10);
x=0;                    y=0;
while(count > 0)
{        obcode[loc].code[x++]=address[y++];
if(x > 1)
{
x=0;
loc++;
count--;
}
}
}
count=0;
n=0;
fprintf(fp3,"%d\t",obcode[0].add);
for(i=0;i<inc;i++)
{
fprintf(fp3,"%s",obcode[i].code);
n++;
if(n>3)
{
fprintf(fp3,"\t");
n=0;
count++;
}
if(count>3)
{
fprintf(fp3,"\n%d\t",obcode[i+1].add);
count=0;
}
}
getch();
}
```

d) Output:

INPUT FILES:

dlink1in.txt

```
H PROGA 000000 000070
D LISTA 000040 ENDA 000054
R LISTB ENDB LISTC ENDC
T 000020 10 03201D 77100004 150014
T 000054 16 100014 15100006 00002F 100014 FFFFC0
M 000024 05 +LISTB
M 000054 06 +LISTC
M 000058 06 +ENDC
M 000064 06 +LISTB
E 000000
H PROGB 000000 000088
D LISTB 000060 ENDB 000070
R LISTA ENDA LISTC ENDC
T 000036 11 03100000 772027 05100000
T 000070 18 100000 05100006 05100020 05100030 100000
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 7100004 05100000
T 000042 15 100030 100008 100011 100000 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
```

dlink1out.txt

```
PROGA       **      2000   70
**      LISTA       2040   0
**      ENDA 2054   0
PROGB       **      2070   88
**      LISTB       2130   0
**      ENDB 2140   0
PROGC       **      2158   57
**      LISTC       2188   0
**      ENDC 2200   0
```

OUTPUT FILE:

Dlink2out.txt

```
2000   xxxxxxxx   xxxxxxxx   xxxxxxxx   xxxxxxxx
2016   xxxxxxxx   03201D77   10213415   0014xxxx
2032   xxxxxxxx   xxxxxxxx   xxxxxxxx   xxxxxxxx
2048   xxxxxxxx   xxxx1022   02151022   0600002F
2064   102144FF   FFC0xxxx   xxxxxxxx   xxxxxxxx
2080   xxxxxxxx   xxxxxxxx   xxxxxxxx   xxxxxxxx
2096   xxxxxxxx   xxxxxxxx   xxxx0310   20407720
2112   27051020   54xxxxxx   xxxxxxxx   xxxxxxxx
2128   xxxxxxxx   xxxxxxxx   xxxxxxxx   10205405
2144   10220605   10222005   10208410   0000xxxx
2160   xxxxxxxx   xxxxxxxx   xxxxxxxx   xxxxxxxx
2176   03102040   77102134   05102054   xxxxxxxx
2192   xxxxxxxx   xxxxxxxx 10003010 00081020
2208   51102054   102130
```

e) Result:

Thus the pass 2 of a direct linking loader is implemented and the result is verified successfully.

VIVA – QUESTION AND ANSWER:

1. What is the function performed in voice input device?

Voice-input devices, which translate spoken words to their textual equivalents, may prove to be the text input devices of the future. Voice recognizers are currently available for command input on some systems.

2. What are called tokens?

The lexical analyzer tracks the source program one character at a time by making the source program into sequence of atomic units is called tokens.

3. Name some of typical tokens.

Identifiers, keywords, constants, operators and punctuation symbols such as commas and parentheses are typical tokens.

4. Mention the main disadvantage of interpreter.

The main disadvantage of interpreter is that the execution time of interpreted program is slower than that of a corresponding compiled object program.

5. What is meant by code optimization?

The code optimization is designed to improve the intermediate code, which helps the object program to run faster and takes less space.

Exercise Number: 10

Title of the Exercise : IMPLEMENTATION OF A SIMPLE TEXT EDITOR
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

- To Create a Component for retrieving stock market exchange information using CORBA

FACILITIES REQUIRED AND PROCEDURE

a) Facilities Required:

S.No.	Facilities required	Quantity
1	System	1
2	O/S	Windows XP
3	S/W name	C or C++

b) Procedure:

Step no.	Details of the step
1	Begin
2	Provide a blank screen for the user to type the document.
3	Instruct the user to enter the text using the editor.
4	Characters displayed on the screen are stored in a character array.
5	If SAVE option is selected a. Get the data file name from the user. b. Open the file in write mode. c. Move the content of the character array to the file and close the file. 1. If OPEN option is selected a. Open the data file in the read mode. b. Read one character at a time from the file and moved it to the character array. c. Repeat the above steps until the end of file is reached. d. Display the character from the character array on the screen. Finally close the text editor and exit.

c) Program:

```
#include<stdio.h>          #include<conio.h>
#include<ctype.h>          #include<dos.h>
#include<iostream.h>       #include<fstream.h>

char filename[15];
char buff[1000];
int curx,cury,count;
```

```
void cur_pos()
{
curx=wherex();
cury=wherey();
textcolor(12);
textbackground(9);
gotoxy(35,20);
cout<<"\n";
cout<<"**********************************************************************\n";
cout<<"\n";
cout<<"                                   TEXT EDITOR               \n";
cout<<"\n";
cout<<"**********************************************************************\n";
cout<<"\n Type ur text and then press escape key\n";
gotoxy(100,100);
cprintf("%2d%2d",curx,cury);
gotoxy(curx,cury);
}
void main()
{
char ch,c;
ofstream outfile;
ifstream infile;
clrscr();
cur_pos();
curx=wherex();
cury=wherey();
while(c!=27)
{
c=getch();
switch(c)
{
case 80:gotoxy(curx,cury+1);
break;
case 77:gotoxy(curx+1,cury);
break;
case 72:gotoxy(curx,cury-1);
break;
case 75:gotoxy(curx-1,cury);
break;
case 32:printf(" ");
buff[count++]=' ';
break;
case 13:gotoxy(1,cury+1);
buff[count++]='\n';
break;
default:textcolor(13);
if((c >= 65) && (c <= 122) || (c >= 48 && c <= 57))
printf("%c",c);
break;
}
buff[count++]=c;
cur_pos();
}
cprintf("\n\nDo u want to save?");
scanf("%c",&c);
if(c == 'y')
{
cprintf("\n\nEnter the filename with extension in 8 char only:    ");
scanf("%s",filename);
outfile.open(filename,ios::out);
outfile<<buff;
outfile.close();
cout<<"\nDo u want to open\n";
ch=getch();
{
if(ch=='y')
{
cprintf("\n\nEnter the filename to open:     ");
scanf("%s",filename);
infile.open(filename,ios::in);
infile.get(buff,count,'*');
gotoxy(90,1);
printf("%s",buff);
getch();
infile.close();
}
}
}
}
```

d) Output:

```
hi welcome

   This is a sample text editor program for ECE data structures lab

                              Do u want to save? y
Enter the filename with extension in 8 char only:     sample.txt

Do u want to open

Enter the filename to open:     sample.txt
hi welcome
         This  is a  sample  text  editor  program for  ECE  data  structures  lab
*******************************************************
               TEXT EDITOR
*******************************************************
```

```
    Type ur text and then press escape key

    28 5
```

e) Result:

Thus the text editor is implemented and the operations are verified successfully.

VIVA – QUESTION AND ANSWER:

1. What is error handler?

The error handler is used to check if there is an error in the program. If any error, it should warn the programmer by instructions to proceed from phase to phase.

2. Name some of text editors.

- line editors
- stream editors
- screen editors
- word processors
- structure editors

3. Mention the features of word processors.

- moving text from one place to another
- merging of text
- searching
- word replacement

4. Define traveling phase.

The phase specifies the region of interest. Traveling is achieved using operations such as next screen, bottom, find pattern.

5. Give the components of editor structure

4 components

- Editing component
- Traveling component
- Viewing component
- Display component

6. What are the basic types of computing environments used in editor's functions?

Editor's function in three basic types of computing environments

i. Time sharing ii. Stand-alone iii. Distributed

Exercise Number: 11

Title of the Exercise : LINE AND SCREEN EDITOR
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

- To write a program to implement line and screen editor

FACILITIES REQUIRED AND PROCEDURE

a) Facilities Required:

S.No.	Facilities required	Quantity
1	System	1
2	O/S	Windows XP
3	S/W name	C or C++

b) Procedure:

Step no	Details of the step
1	Initialize all the variables.
2	Design a menu through which we can create a symbol Table and perform operations as insert, modify, search and display.
3	Create a symbol table with fields as variable and value using create option. Entries may be added to the table while it is created itself.
4	Append new contents to the symbol table with the constraint that there is no duplication of entries, using insert option.
5	Modify existing content of the table using modify option.
6	Use display option to display the contents of the table.
7	End of the program.

c) Program:

```
 #include<stdio.h>

#include<conio.h>
#include<graphics.h>
void main()
{
int i,j,k,q;
clrscr();
textattr(10);
gotoxy(1,1);
cprintf("LINE AND SCREEN EDITOR\n\n");
gotoxy(5,5);
cprintf("at 5,5");
gotoxy(20,5);
cprintf("at 5,20");
gotoxy(5,7);
cprintf("at 7,5");
gotoxy(20,7);
cprintf("at 7,20");
gotoxy(5,9);
cprintf("at 9,5");
gotoxy(20,9);
cprintf("at 9,20");
for(j=0;j<q;j+2)
{
gotoxy(30,i);
cprintf("at %d 30",i);
gotoxy(40,i);
cprintf("at %d 40",i);
}
getch();
}
```

d) Output:

```
   LINE AND SCREEN EDITOR

at 5,5          at 5,20
at 7,5          at 7,20
at 9,5          at 9,20
```

e) Result:

Thus the line and screen editor is created and operations are verified successfully.

VIVA – QUESTION AND ANSWER:

1. What is system software?
System software consists of variety of programs that supports the operations of a computer. This makes it possible for the user to focus on an application or other problem to be solved, without needing to know the details of how the machine works internally.Examples of system software are text-editors, compilers, loaders or linkers, debuggers, assemblers, and operating systems.

2. Give some applications of operating system.
- to make the computer easier to use
- to manage the resources in computer
- process management
- data and memory management
- to provide security to the user.
- Operating system acts as an interface between the user and the system
  Eg: windows, Linux, UNIX, dos

3. What is SIC machine?
SIC refers to Simplified Instruction Computer which is a hypothetical computer that has been designed to include the hardware features most often found on real machines, while avoiding unusual and irrelevant complexities. This allows to clearly separating the central concepts of system software from the implementation details associated with a particular machine

Exercise Number: 12

Title of the Exercise : BOOTSTRAP LOADERS
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT
To write a program to implement bootstrap loaders .
FACILITIES REQUIRED AND PROCEDURE

a) Facilities Required:

S.No.	Facilities required	Quantity
1	System	1
2	O/S	Windows XP
3	S/W name	C or C++

b) Procedure:

Step no	Details of the step
1	Initialize all the variables.
2	Design a menu through which we can create a symbol Table and perform operations as insert, modify, search and display.
3	Include the necessary header files and variable.
4	Open the three files a.f1=macin.dat with read privilege b. f2=macout.dat with write privilege c. f3= deftab.dat with write privilege
5	Get the variable form f1 file macin.dat for label,opcode,operand
6	Read the variable until the opcode is not is equal to zero
7	End of the program.

c) Program:

```
 #include<stdio.h>              #include<string.h>
char line[5][20];              void numtok();
int cur=0;             int lth;
void main()
{
FILE *fin;
int i=0,m=0,num=0,p=0,k=0,l,j;                  int prgstart,prglen,textaddr,disp;
char str[70];
struct loccontent
{
int addr;          char content[3];
}mem[200];
clrscr();
for(i=0;i<200;i++)
{
mem[i].content[0]='x';          mem[i].content[1]='x';          mem[i].content[2]='/0';
}
fin=fopen("loadin.txt","r");
```

```
while(!feof(fin))
{
fgets(str,70,fin);                    numtok(str);
if(strcmp(line[0],"H")==0)
{
printf("Program name is%s\n",line[1]);                    prgstart=atoi(line[2]);
prglen=atoi(line[3]);               j=0;
for(i=prgstart;i<=prgstart+prglen;i++)
{
mem[j].addr=i;                      j ++;
}
}
else if(strcmp(line[0],"T")==0)
{
textaddr=atoi(line[1]);                       disp=textaddr-prgstart;
for(m=3;m<lth;++m)
{
for(i=0;i<=5;i+=2)
{
mem[disp].addr=textaddr++;                        mem[disp].content[0]=line[m][i];
mem[disp].content[1]=line[m][i+1];              mem[disp].content[2]='\0';
disp++;
}        }         }
else if(strcmp(line[0],"E")==0)
break;
k=0;
}
for(i=0;i<prglen;i+=16)
{printf("%d\t",mem[k].addr);
for(l=0;l<4;l++)
{for(j=0;j<4;j++)
{printf("%c%c",mem[k].content[0],mem[k].content[1]);
k++;
}
printf("\t");
}
printf("\n");
}
getch();
}
void numtok(char str[70])
{
int i=0,j=0,k=0;
while(str[j]!='\0')
{line[i][k]=str[j];
if(str[j]=='^')
{line[i][k]='\0';
k=-1;
i++;
}
k++;
j++;
}
line[i][k]='\0';
lth=i+1;
}
```

d) Output:

INPUT FILE:

```
loadin.txt:
H^COPY^1000^115
T^1000^12^141003^482039^001036^281030
T^1012^9^345890^581056^454645
T^1012^12^141033^482049^001036^281030
E^1000
```

OUTPUT FILE:

```
Program name isCOPY
1000    14100348    20390010    36281030    14103348
1016    20490010    36281030    xxxxxxxx    xxxxxxxx
1032    xxxxxxxx    xxxxxxxx    xxxxxxxx    xxxxxxxx
1048    xxxxxxxx    xxxxxxxx    xxxxxxxx    xxxxxxxx
1064    xxxxxxxx    xxxxxxxx    xxxxxxxx    xxxxxxxx
1080    xxxxxxxx    xxxxxxxx    xxxxxxxx    xxxxxxxx
1096    xxxxxxxx    xxxxxxxx    xxxxxxxx    xxxxxxxx
1112    xxxxxxxx    xxxxxxxx    xxxxxxxx    xxxxxxxx
```

e) Result:

Thus the bootstrap loaders is created and operations are verified successfully

VIVA – QUESTION AND ANSWER:

1. What is system software?

System software consists of variety of programs that supports the operations of a computer. This makes it possible for the user to focus on an application or other problem to be solved, without needing to know the details of how the machine works internally.Examples of system software are text-editors, compilers, loaders or linkers, debuggers, assemblers, and operating systems.

2. Give some applications of operating system.

- to make the computer easier to use to manage the resources in computer
- process management data and memory management
- to provide security to the user.
- Operating system acts as an interface between the user and the system
  Eg: windows, Linux, UNIX, dos

3. What is SIC machine?

SIC refers to Simplified Instruction Computer which is a hypothetical computer that has been designed to include the hardware features most often found on real machines, while avoiding unusual and irrelevant complexities. This allows to clearly separating the central concepts of system software from the implementation details associated with a particular machine

Exercise Number: 13

Title of the Exercise : MULTI PASS ASSEMBLERS
Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To write a program to implement symbol table.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities Required:

S.No.	Facilities required	Quantity
1	System	1
2	O/S	Windows XP
3	S/W name	C or C++

b) Procedure:

Step no	Details of the step
1	Initialize all the variables.
2	write line to intermediate file
3	read next input line
4	Append new contents to the symbol table with the constraint that there is no duplication of entries, using insert option.
5	Modify existing content of the table using modify option.
6	Use display option to display the contents of the table.
7	End of the program.

c) Program:

```
 #include<stdlib.h>
#include<conio.h>
#include<stdio.h>
#include<string.h>
int search(char sym[10]);
char praddr[10];
int k,data[100],mem[100],i=0,q,d=0, value,j;
int csaddr,exaddr,cslth;
char type[10],addr[10],obj[10],mflag[10],op;
void main()
{
FILE *in,*address;
in=fopen("OBJECT.TXT","r");
address=fopen("ADDRESS.TXT","r");
clrscr();
fscanf(address,"%s",praddr);
fclose(address);
csaddr=atoi(praddr);
```

```
exaddr=atoi(praddr);
fscanf(in,"%s\t%s\t%s\t%s\t",type,addr,obj,mflag);
while(!feof(in))
{
cslth=atoi(obj);
while(strcmp(type,"E")!=0)
{
fscanf(in,"%s\t%s\t%s\t%s\t",type,addr,obj,mflag);
if(strcmp(type,"T")==0)
{
mem[i]=atoi(addr)+csaddr;
data[i]=atoi(obj);
i++;
}
else if(strcmp(type,"M")==0)
{
op=mflag[0];
for(j=1;j<strlen(mflag);j++)
mflag[j-1]=mflag[j];
mflag[j-1]='\0';
value=search(mflag);
q=d+atoi(addr)/3;
if(op=='+')
data[q]=data[q]+value;
else
data[q]=data[q]-value;
}
}
d=cslth/3;
exaddr=csaddr+atoi(addr);
csaddr=csaddr+cslth;
fscanf(in,"%s\t%s\t%s\t%s\n",type,addr,obj,mflag);
}
fclose(in);
q=(exaddr-(atoi(praddr)))/3;
printf("Program Loaded in memory is ready for execution\n");
printf("Execution Starting Address:%d\n",exaddr);
printf("ADDRESS\t DATA\n");
for(j=0;j<i;j++)
printf("%d\t%d\n",mem[j],data[j]);
gotoxy(1,q+4);
getch();
}
int search(char sym[10])
{
FILE *fp;
char s1[10],s2[10],s3[10],s4[10];
fp=fopen("ESTAB.TXT","r");
while(!feof(fp))
{
```

```
fscanf(fp,"%s\t%s\t%s\t%s\n",s1,s2,s3,s4);
if(strcmp(s2,sym)==0)
{
fclose(fp);
return(atoi(s3));
}
}
fclose(fp);
return 0;
}
```

d) Output:

```
 Program Loaded in memory is ready for execution
Execution Starting Address:9009
ADDRESS  DATA
9000   6
9003   10006
9006   9
9009   6
9012   10006
9015   9
```

e) Result:

Thus the multi pass assemblers is created and operations are verified successfully.

VIVA – QUESTION AND ANSWER:

1. What is system software?

System software consists of variety of programs that supports the operations of a computer. This makes it possible for the user to focus on an application or other problem to be solved, without needing to know the details of how the machine works internally.
Examples of system software are text-editors, compilers, loaders or linkers, debuggers, assemblers, and operating systems.

2. What is SIC machine?

SIC refers to Simplified Instruction Computer which is a hypothetical computer that has been designed to include the hardware features most often found on real machines, while avoiding unusual and irrelevant complexities. This allows to clearly separating the central concepts of system software from the implementation details associated with a particular machine

3. What is the important machine structures used in the design of system software?

- Memory structure Registers Data formats
- Instruction formats Addressing modes Instruction set

4. Define compiler and interpreter.

Compiler is a set of program which converts the whole high level language program to machine language program. Interpreter is a set of programs which converts high level language program to machine language program line by line.

CS2308 - SYSTEM SOFTWARE LAB

MODEL QUESTION SET

1. Write a C program to implement a symbol table with functions to create, insert, modify, search, and display.
2. Write a C program to implement pass one of a two pass assembler.
3. Write a C program to implement pass two of a two pass assembler.
4. Write a C program to implement a single pass assembler.
5. Write a C program to implement a two pass macro processor
6. Write a C program to implement a single pass macro processor.
7. Write a C program to implement an absolute loader.
8. Write a C program to implement a relocating loader.
9. Write a C program to implement pass one of a direct-linking loader.
10. Write a C program to implement pass two of a direct-linking loader.
11. Write a C++ program to implement a simple text editor with features like insertion / deletion of a character, word, and sentence.
12. Write a C program to implement a symbol table with suitable hashing