Fr. Conceicao Rodrigues College of Engineering

Department of Computer Engineering

Academic Term: Jan-Apr 2022

System Programming & Compiler Construction

Department of Computer

EngineeringAcademic

Term: Jan-Apr 2022

Class : T.E Computer Sem -VII
Subject: System Programming and Compiler Construction

Practical No:	**1**
Title:	Implement Symbol Table.
Date of Performance:	28/01/2022
Date of Submission:	28/01/2022
Roll No:	**8940**
Name of the Student:	**Warren Fernandes**

Evaluation:

Sr. No	Rubric	Grade
1	On time Completion & Submission (2)	
2	Output (3)	
3	Code Optimization (3)	
4	Knowledge of the topic (2)	
5	Total (10)	

Signature of the Teacher:

WARREN FERNANDES **TE COMPS B (8940)**

Lab 1: SPCC - C program for implementing Symbol Table

CODE

```
# include <stdio.h>
# include <string.h>
# define null 0
int size=0;
void insert();
void del();
int search(char lab[]);
void modify();
void display();
struct symbtab
{
        char label[10];
        int addr;
        struct symtab *next;
};
struct symbtab *first,*last;
void main()
{
        int op;
        int y;
        char la[10];
        do
        {
                printf("\nSYMBOL TABLE IMPLEMENTATION\n");
                printf("1. INSERT\n");
                printf("2. DISPLAY\n");
                printf("3. DELETE\n");
                printf("4. SEARCH\n");
                printf("5. MODIFY\n");
                printf("6. END\n");
                printf("Enter your option : ");
                scanf("%d",&op);
                switch(op)
                {
                        case 1:
                        insert();
                        display();
                        break;
                        case 2:
                        display();
                        break;
                        case 3:
                        del();
                        display();
                        break;
                        case 4:
                        printf("Enter the label to be searched : ");
                        scanf("%s",la);
                        y=search(la);
                        if(y==1)
                        {
                        printf("The label is already in the symbol Table");
                        }
                        else
                        {
                        printf("The label is not found in the symbol table");
                        }
                        break;
                        case 5:
                        modify();
                        display();
                        break;
                        case 6:
                        break;
                }

        }
        while(op<6);
}
void insert()
{
        int n;
        char l[10];
        printf("Enter the label : ");
        scanf("%s",l);
        n=search(l);
        if(n==1)
        {
        printf("The label already exists. Duplicate cant be inserted\n");
        }
        else
        {
                struct symbtab *p;
                p=malloc(sizeof(struct symbtab));
                strcpy(p->label,l);
                printf("Enter the address : ");
                scanf("%d",&p->addr);
                p->next=null;
                if(size==0)
                {
                        first=p;
                        last=p;
                }
                else
                {
                        last->next=p;
                        last=p;
                }
                size++;
        }

}
void display()
{
        int i;
        struct symbtab *p;
        p=first;
        printf("LABEL\tADDRESS\n");
        for(i=0;i<size;i++)
        {
                printf("%s\t%d\n",p->label,p->addr);
                 p=p->next;
        }

}
int search(char lab[])
{
        int i,flag=0;
        struct symbtab *p;
        p=first;
        for(i=0;i<size;i++)
        {
                if(strcmp(p->label,lab)==0)
                {
                        flag=1;
                }
                p=p->next;
        }
        return flag;
}
void modify()
{
        char l[10],nl[10];
        int add, choice, i, s;
        struct symbtab *p;
        p=first;
        printf("What do you want to modify?\n");
        printf("1. Only the label\n");
        printf("2. Only the address of a particular label\n");
        printf("3. Both the label and address\n");
        printf("Enter your choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
                case 1:
                printf("Enter the old label\n");
```

```
scanf("%s",l);
printf("Enter the new label\n");
scanf("%s",nl);
s=search(l);
if(s==0)
{
        printf("NO such label");
}
else
{
        for(i=0;i<size;i++)
        {
                if(strcmp(p->label,l)==0)
                {
                        strcpy(p->label,nl);
                }
                p=p->next;
        }
}
break;
case 2:
printf("Enter the label whose address is to modified\n");
scanf("%s",l);
printf("Enter the new address\n");
 scanf("%d",&add);
s=search(l);
if(s==0)
{
        printf("NO such label");
}
else
{
        for(i=0;i<size;i++)
        {
                if(strcmp(p->label,l)==0)
                {
                        p->addr=add;
                }
                p=p->next;
        }
}
break;
case 3:
printf("Enter the old label : ");
scanf("%s",l);
printf("Enter the new label : ");
scanf("%s",nl);
printf("Enter the new address : ");
scanf("%d",&add);
s=search(l);
if(s==0)
```

```
            {
                    printf("NO such label");
            }
            else
            {
                    for(i=0;i<size;i++)
                    {
                            if(strcmp(p->label,l)==0)
                            {
                                    strcpy(p->label,nl);
                                    p->addr=add;
                            }
                            p=p->next;
                    }
            }
            break;
    }
}
void del()
{
    int a;
    char l[10];
    struct symbtab *p,*q;
    p=first;
    printf("Enter the label to be deleted\n");
    scanf("%s",l);
    a=search(l);
    if(a==0)
    {
            printf("Label not found\n");
    }
    else
    {
            if(strcmp(first->label,l)==0)
            {
                    first=first->next;
            }
            else if(strcmp(last->label,l)==0)
            {
                    q=p->next;
                    while(strcmp(q->label,l)!=0)
                    {
                            p=p->next;
                            q=q->next;
                    }
                    p->next=null;
                    last=p;
            }
            else
            {
                    q=p->next;
```

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

OUTPUT:

```
SYMBOL TABLE IMPLEMENTATION
1. INSERT
2. DISPLAY
3. DELETE
4. SEARCH
5. MODIFY
6. END
Enter your option : 1
Enter the label : A
Enter the address : 10
LABEL   ADDRESS
A       10

SYMBOL TABLE IMPLEMENTATION
1. INSERT
2. DISPLAY
3. DELETE
4. SEARCH
5. MODIFY
6. END
Enter your option : 2
LABEL   ADDRESS
A       10

SYMBOL TABLE IMPLEMENTATION
1. INSERT
2. DISPLAY
3. DELETE
4. SEARCH
5. MODIFY
6. END
Enter your option : 1
Enter the label : B
Enter the address : 11
LABEL   ADDRESS
A       10
B       11
```

```
SYMBOL TABLE IMPLEMENTATION
1. INSERT
2. DISPLAY
3. DELETE
4. SEARCH
5. MODIFY
6. END
Enter your option : 2
LABEL   ADDRESS
A       10
B       11

SYMBOL TABLE IMPLEMENTATION
1. INSERT
2. DISPLAY
3. DELETE
4. SEARCH
5. MODIFY
6. END
Enter your option : 3
Enter the label to be deleted
B
LABEL   ADDRESS
A       10

SYMBOL TABLE IMPLEMENTATION
1. INSERT
2. DISPLAY
3. DELETE
4. SEARCH
5. MODIFY
6. END
Enter your option : 2
LABEL   ADDRESS
A       10

SYMBOL TABLE IMPLEMENTATION
1. INSERT
2. DISPLAY
3. DELETE
4. SEARCH
5. MODIFY
6. END
Enter your option : 1
Enter the label : C
Enter the address : 7
LABEL   ADDRESS
A       10
C       7

SYMBOL TABLE IMPLEMENTATION
```

```
1. INSERT
2. DISPLAY
3. DELETE
4. SEARCH
5. MODIFY
6. END
Enter your option : 2
LABEL   ADDRESS
A       10
C       7

SYMBOL TABLE IMPLEMENTATION
1. INSERT
2. DISPLAY
3. DELETE
4. SEARCH
5. MODIFY
6. END
Enter your option : 4
Enter the label to be searched : A
The label is already in the symbol Table
SYMBOL TABLE IMPLEMENTATION
1. INSERT
2. DISPLAY
3. DELETE
4. SEARCH
5. MODIFY
6. END
Enter your option : 5
What do you want to modify?
1. Only the label
2. Only the address of a particular label
3. Both the label and address
Enter your choice : 2
Enter the label whose address is to modified
A
Enter the new address
12
LABEL   ADDRESS
A       12
C       7

SYMBOL TABLE IMPLEMENTATION
1. INSERT
2. DISPLAY
3. DELETE
4. SEARCH
5. MODIFY
6. END
Enter your option : 6
```

Department of Computer

EngineeringAcademic

Term: Jan-Apr 2022

Class : T.E Computer Sem -VII
Subject: System Programming and Compiler Construction

Practical No:	**2**
Title:	Implement basic pass1 and pass2 of two pass assemblers.
Date of Performance:	04/02/2022
Date of Submission:	04/02/2022
Roll No:	**8940**
Name of the Student:	**Warren Fernandes**

Evaluation:

Sr. No	**Rubric**	**Grade**
1	**On time Completion & Submission (2)**	
2	**Output (3)**	
3	**Code Optimization (3)**	
4	**Knowledge of the topic (2)**	
5	**Total (10)**	

Signature of the Teacher:

Warren Fernandes TE COMPS B (8940) BATCH B

Experiment No 2

Aim: Write a program to implement Two Pass Assembler

Learning Objective: Translating mnemonic operation codes to their machine language equivalents. Assigning machine addresses to symbolic labels used by the programmer. Lastly to convert assembly language to binary.

Algorithm:

<u>Pass 1:</u>

```
1. Start
2.Intialize location counter to zero

3.Read opcode field of next instruction.

4.search opcode in pseudo opcode Table(POT)

5.If opcode is found in POT

        5.1 If it is 'DS' or 'DC'

            Adjust location counter to proper alignment.

            Assign length of data field to 'L'

            Go to step 9

        5.2 If it is 'EQU'

                Evaluate operand field

                Assign values to symbol in label field

                Go to step 3

        5.3 If it is 'USING' or 'DROP' Go to step 3
        5.4 If it is 'END'

              Assign value to symbol in label field

              Go to step 3
```

```
6.   Search opcode in Machine Opcode Table.

7.   Assign its length to ‘L’.
8.   Process any literals and enter them into literal
Table.
9.If symbol is there in the label field

        Assign current value of Location Counter to symbol

10. Location Counter= Location Counter +L.

11.Go to step 3.
12. Stop.
```

Pass2:

```
1.Start
2.   Intialize location counter to zero.
3.   Read opcode field of next instruction.
4.   Search opcode in pseudo opcode table.

5.   If opcode is found in pseudo opcode Table

     5.1 If it is ‘DS’ or ‘DC’

             Adjust location counter to proper alignment.

             If it is ‘DC’ opcode form constant and insert in assembled program

             Assign length of data field to ‘L’

             Go to step 6.4
     5.2 If it is ‘EQU’ or ‘START’ ignore it. Go to step 3

     5.3 If it is ‘USING’

         Evaluate operand and enter base reg no. and value into base table

         Go to step 3

     5.4 If it is ‘DROP’

         Indicate base reg no . available in base table . Go to step 3

     5.5 If it is ‘END’

         Generate literals for entries in Literal Table
```

Go to step 12

6 Search opcode in MOT

7. Get opcode byte and format code

8. Assign its length to ‘L’.
9. Check type of instruction.

10.If it is type ‘RR’ type

10.1 Evaluate both register expressions and insert into second byte.
10.2 Assemble instruction

10.3 Location Counter= Location Counter +L.

10. 4.Go to step 3.

11. If it is ‘RX’ type

11.1 Evaluate register and index expressions and insert into second byte.

11.2 Calculate effective address of operand.
11.3 Determine appropriate displacement and base register
11.4 Put base and displacement into bytes 3 and 4

11.5 Location Counter= Location Counter +L.

11.6 Go to step 11.2 13 Stop.

Implementation Details

1. Read Assembly language input file

2. Display output of Pass1 as the output file with Op-code Table, Symbol Table

3. Display output of pass2 as the Op-code Table, Symbol Table , Copy file

Test Cases:

1 Input symbol which is not defined

2 Input Opcode which is not entered in MOT

Conclusion: Two pass assembler has been implemented

REFERENCE:

PASS 1: https://youtu.be/esDnuGD6kb0

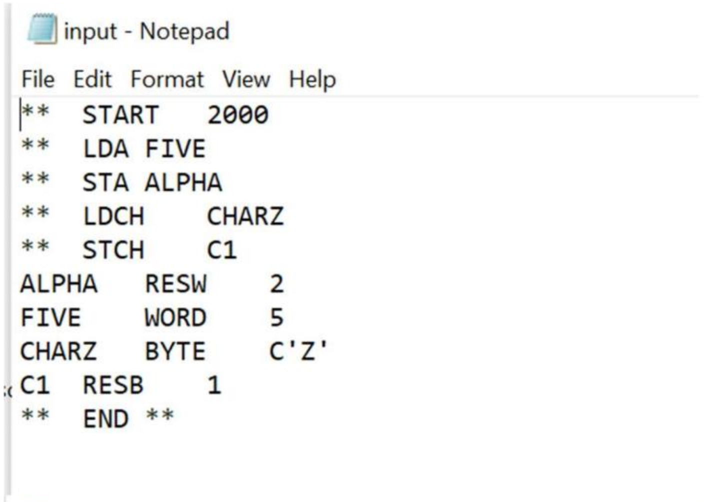
PASS 2: https://forgetcode.com/c/104-pass-two-of-a-two-pass-assembler

PASS 1 CODE

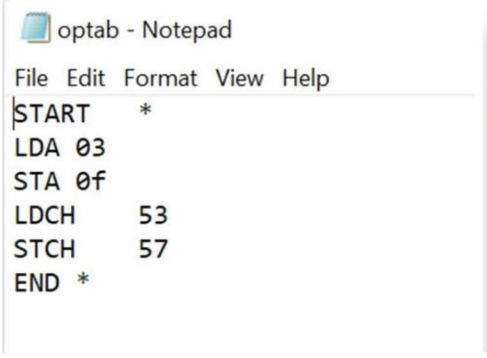
```
#include<stdio.h>
#include<string.h>
#include<stdlib.h> void
main() {
   char opcode[10], operand[10], label[10], code[10], mnemonic[10];     int
locctr, start, length;

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

   fp1 = fopen("input.txt", "r");
fp2 = fopen("optab.txt", "r");     fp3
= fopen("symtbl.txt", "w");
   fp4 = fopen("output.txt", "w");

   fscanf(fp1, "%s\t%s\t%s", label, opcode, operand);

   if(strcmp(opcode, "START")==0) {
start = atoi(operand);
       locctr = start;
       fprintf(fp4, "\t%s\t%s\t%s\n", label, opcode, operand);
       fscanf(fp1, "%s\t%s\t%s", label, opcode, operand);
```

```
    }     else {
locctr = 0;
    }

    while(strcmp(opcode, "END")!=0) {
fprintf(fp4, "%d\t", locctr);

        if(strcmp(label, "**")!=0) {
            fprintf(fp3, "%s\t%d\n", label, locctr);
        }

        fscanf(fp2, "%s\t%s", code, mnemonic);

        while(strcmp(code, "END")!=0) {
if(strcmp(opcode, code)==0) {
                locctr+=3;
break;
            }
            fscanf(fp2, "%s\t%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(fp4, "%s\t%s\t%s\t\n", label, opcode, operand);          fscanf(fp1,
"%s\t%s\t%s", label, opcode, operand);
    }

    fprintf(fp4, "%d\t%s\t%s\t%s\n", locctr, label, opcode, operand);

    length = locctr-start;

    printf("The length of the code : %d\n", length);

    fclose(fp1);
fclose(fp2);     fclose(fp3);
fclose(fp4);
}
```

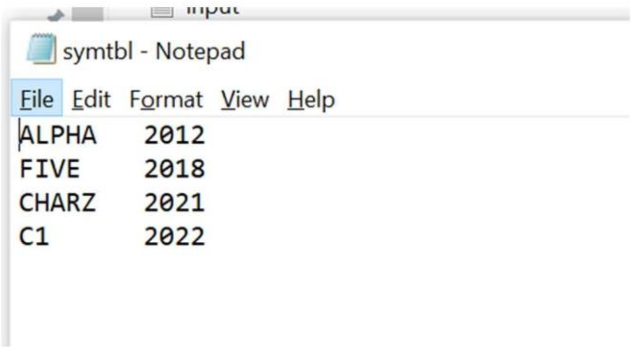
INPUT



```
** START   2000
** LDA FIVE
** STA ALPHA
** LDCH    CHARZ
** STCH    C1
ALPHA   RESW    2
FIVE    WORD    5
CHARZ   BYTE    C'Z'
C1  RESB    1
** END **
```

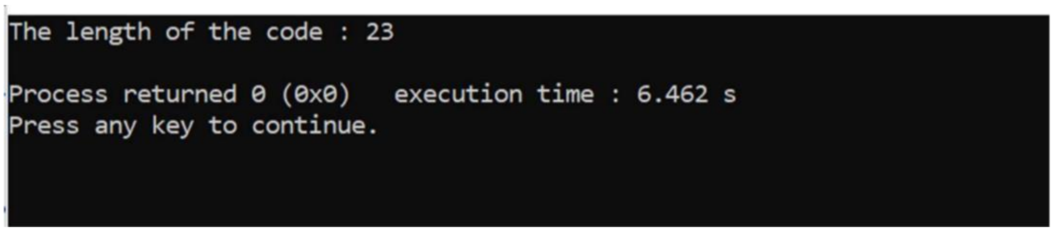


```
START   *
LDA 03
STA 0f
LDCH    53
STCH    57
END *
```

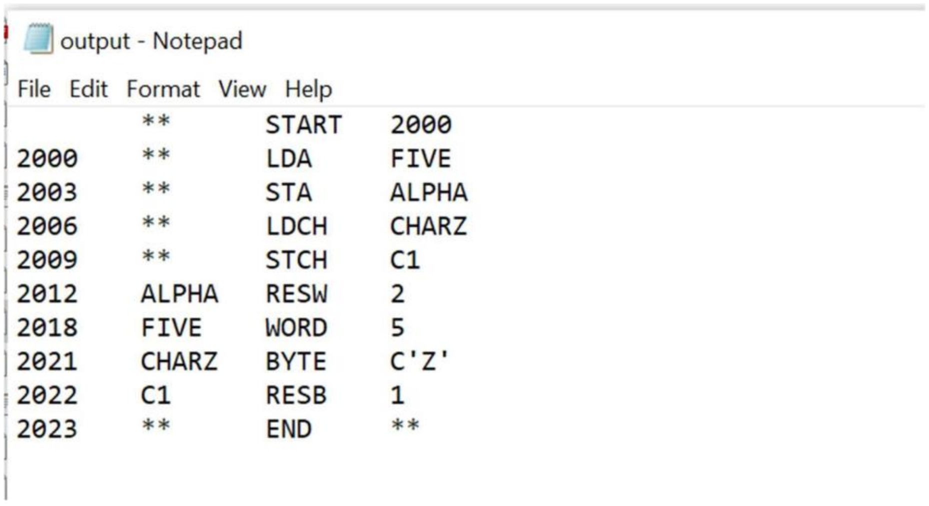


```
ALPHA   2012
FIVE    2018
CHARZ   2021
C1      2022
```

OUTPUT



```
The length of the code : 23

Process returned 0 (0x0)   execution time : 6.462 s
Press any key to continue.
```

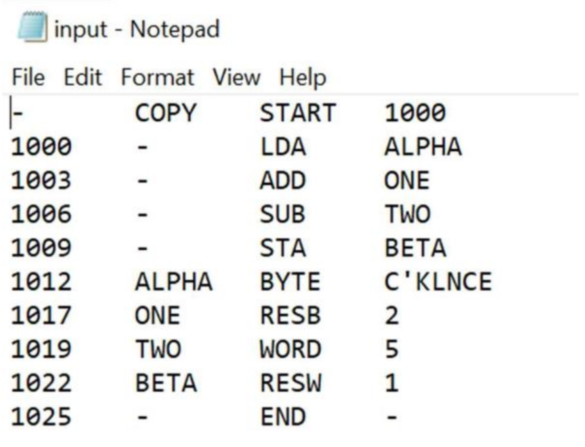


```
        **      START   2000
2000    **      LDA     FIVE
2003    **      STA     ALPHA
2006    **      LDCH    CHARZ
2009    **      STCH    C1
2012    ALPHA   RESW    2
2018    FIVE    WORD    5
2021    CHARZ   BYTE    C'Z'
2022    C1      RESB    1
2023    **      END     **
```

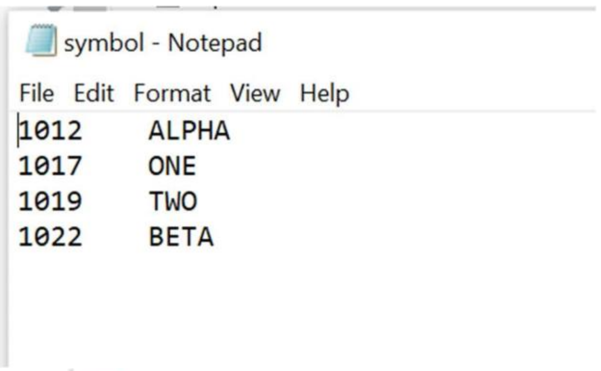
PASS 2 CODE

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<ctype.h> main()
{
FILE  *fint,*ftab,*flen,*fsym,  *fout;
int     op1[10],txtlen,txtlen1,i,j=0,len;
char
add[5],symadd[5],op[5],start[10],temp[30],line[20],label[20],mne[10],operand[10],symtab[10],opm ne[10];

fint=fopen("input.txt","r"); flen=fopen("length.txt","r");
ftab=fopen("optab.txt","r");
fsym=fopen("symbol.txt","r");
fout=fopen("output.txt","w");
fscanf(fint,"%s%s%s%s",add,label,mne,operand);
if(strcmp(mne,"START")==0)
{
strcpy(start,operand);
fscanf(flen,"%d",&len);
}
printf("H^%s^%s^%d\nT^00%s^",label,start,len,start);
fscanf(fint,"%s%s%s%s",add,label,mne,operand);
while(strcmp(mne,"END")!=0)
{
fscanf(ftab,"%s%s",opmne,op);
while(!feof(ftab))
{
if(strcmp(mne,opmne)==0)
{
fclose(ftab); fscanf(fsym,"%s%s",symadd,symtab);
while(!feof(fsym))
{
if(strcmp(operand,symtab)==0)
{
printf("%s%s^",op,symadd);
break;
}
else fscanf(fsym,"%s%s",symadd,symtab);
}
break;
}
else fscanf(ftab,"%s%s",opmne,op);
}
if((strcmp(mne,"BYTE")==0)||(strcmp(mne,"WORD")==0))
{
if(strcmp(mne,"WORD")==0)
printf("0000%s^",operand);
else
{
len=strlen(operand);
for(i=2;i<len;i++)
{
printf("%d",operand[i]);
}
printf("^");
}
}
fscanf(fint,"%s%s%s%s",add,label,mne,operand);
ftab=fopen("optab.txt","r");
fseek(ftab,SEEK_SET,0);
}
printf("\nE^00%s",start);
fclose(fint); fclose(ftab);
fclose(fsym); fclose(flen);
fclose(fout);
getch();
}
```

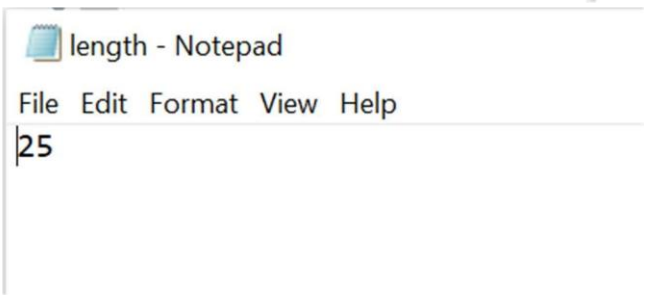
INPUT



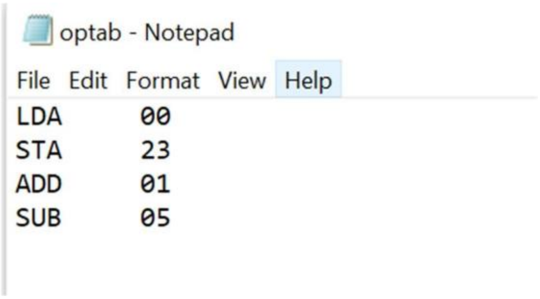
```
-       COPY    START   1000
1000    -       LDA     ALPHA
1003    -       ADD     ONE
1006    -       SUB     TWO
1009    -       STA     BETA
1012    ALPHA   BYTE    C'KLNCE
1017    ONE     RESB    2
1019    TWO     WORD    5
1022    BETA    RESW    1
1025    -       END     -
```



```
1012    ALPHA
1017    ONE
1019    TWO
1022    BETA
```

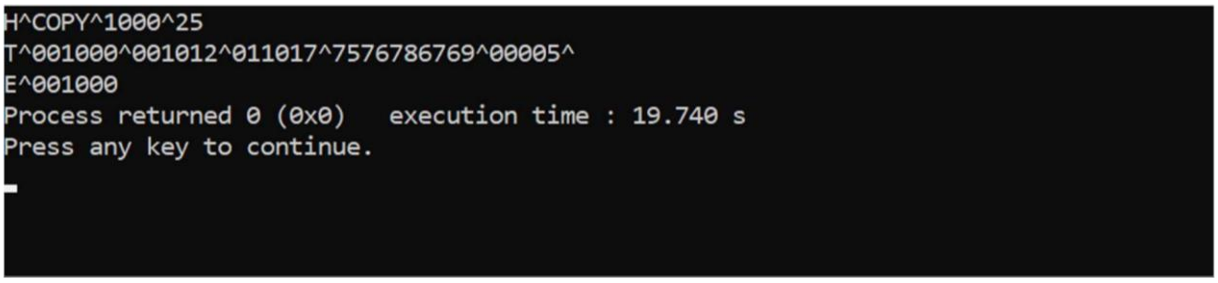


```
25
```



```
LDA     00
STA     23
ADD     01
SUB     05
```

OUTPUT



```
H^COPY^1000^25
T^001000^001012^011017^7576786769^00005^
E^001000
Process returned 0 (0x0)   execution time : 19.740 s
Press any key to continue.
```

Department of Computer

EngineeringAcademic

Term: Jan-Apr 2022

Class : T.E Computer Sem -VII
Subject: System Programming and Compiler Construction

Practical No:	**3**
Title:	Implement basic pass1 and pass2 of two pass Macro Processor.
Date of Performance:	14/02/2022
Date of Submission:	14/02/2022
Roll No:	**8940**
Name of the Student:	**Warren Fernandes**

Evaluation:

Sr. No	Rubric	Grade
1	On time Completion & Submission (2)	
2	Output (3)	
3	Code Optimization (3)	
4	Knowledge of the topic (2)	
5	Total (10)	

Signature of the Teacher:

IMPLEMENT A TWO PASS MACRO PROCESSOR

AIM:

To implement two pass macro processor using in C language.

ALGORITHM:

1. Start the program execution.

2. Macro instructions are included in a separate file.

3. The instructions with ‘macro’,’mend’,’call’ on them should not be printed in the output.

4. Print all other instructions such as start,load,store,add,sub Etc with their values.

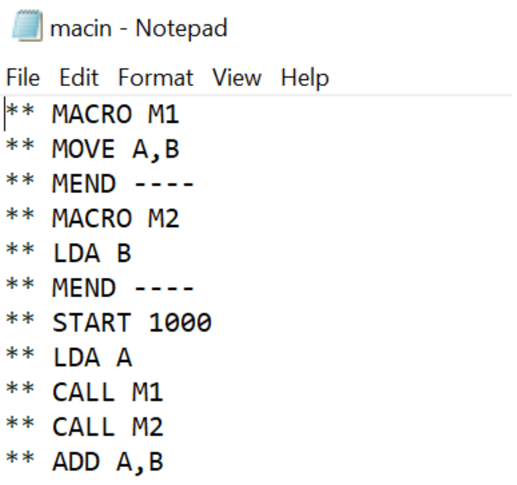
5. Stop the program execution.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.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("outm.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",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);
}
```

Input:

macin.txt

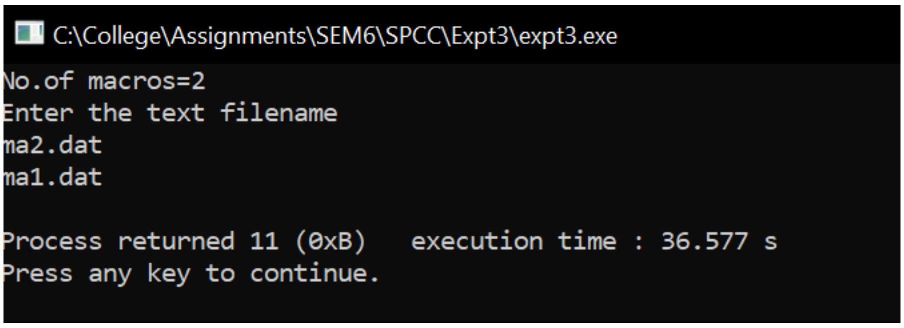


macin - Notepad

File Edit Format View Help

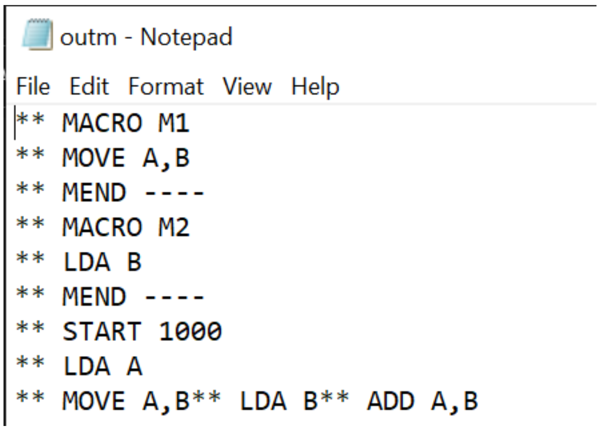
```
** MACRO M1
** MOVE A,B
** MEND ----
** MACRO M2
** LDA B
** MEND ----
** START 1000
** LDA A
** CALL M1
** CALL M2
** ADD A,B
```

OUTPUT:



C:\College\Assignments\SEM6\SPCC\Expt3\expt3.exe

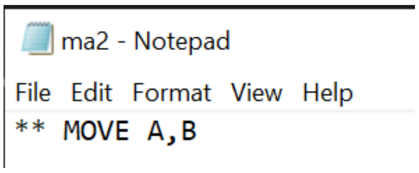
```
No.of macros=2
Enter the text filename
ma2.dat
ma1.dat

Process returned 11 (0xB)   execution time : 36.577 s
Press any key to continue.
```



outm - Notepad

File Edit Format View Help

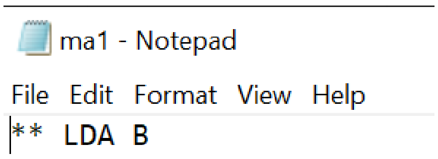
```
** MACRO M1
** MOVE A,B
** MEND ----
** MACRO M2
** LDA B
** MEND ----
** START 1000
** LDA A
** MOVE A,B** LDA B** ADD A,B
```



ma2 - Notepad

File Edit Format View Help

```
** MOVE A,B
```



ma1 - Notepad

File Edit Format View Help

```
** LDA B
```

RESULT:

Thus, a two pass macro processor is implemented successfully using in C language.

Ex.No:6 IMPLEMENT A SINGLE PASS MACRO PROCESSOR

AIM:

To implement a single pass macro processor using in C language.

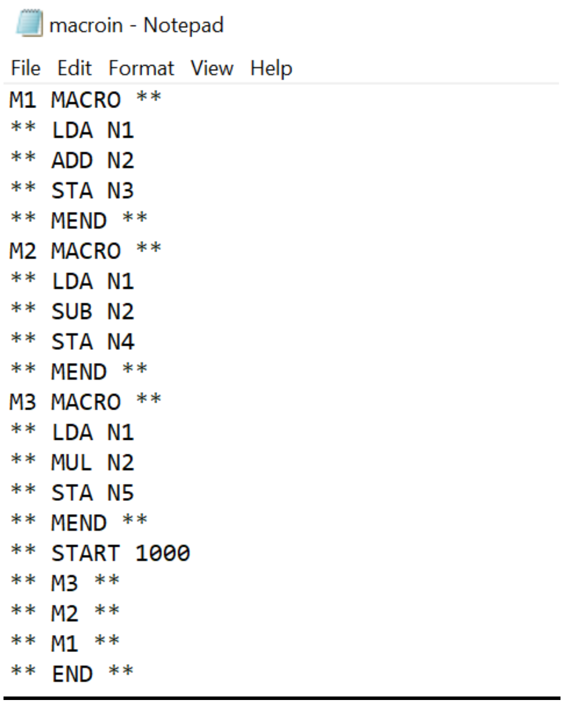
ALGORITHM:

```
STEP 1:  GET THE STATEMENT FROM THE INPUT FILE

STEP 2:  IF THE STATEMENT HAS THE DIRECTIVE "MACRO", THEN THE
         NUMBER OF MACRO "N" WILL BE INCREMENTED BY 1

STEP 3:  REPEAT THE STEPS 1 AND 2 UNTIL AN END OF FILE IS
         ENCOUNTERED

STEP 4:  OPEN "N" NUMBER OF MACRO FILES IN WRITE MODE AND REWIND THE
         INPUT FILE POINTER

STEP 5:  IF THE DIRECTIVE IS "MACRO" THEN, DO THE FOLLOWING

    STEP 5.1: ENTER THE MACRO NAME PRESENT IN THE OPERAND FIELD

    STEP 5.2: WRITE THE LINE TO THE EXPANDED OUTPUT FILE

    STEP 5.3: ENTER THE LINES IN THE BODY OF EACH MACRO IN TO
              THE CORRESPONDING FILES ALREADY OPENED IN STEP 4.

    STEP 5.4: WRITE THE BODY OF EACH MACRO TO THE EXPANDED
              OUTPUT FILE UNTIL A "MEND" IS REACHED

STEP 6:  WRITE THE REMAINING LINES DIRECTLY TO THE EXPANDED FILE.
```

PROGRAM:

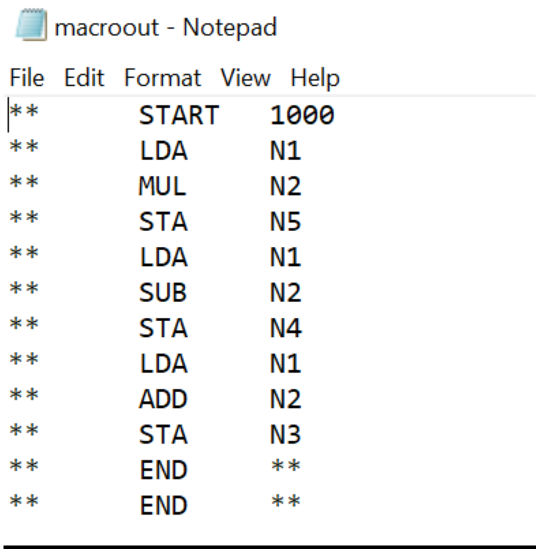
```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
void main()
{
 int n,flag,i;
 char ilab[20],iopd[20],oper[20],NAMTAB[20][20];
 FILE *fp1,*fp2,*DEFTAB;
 clrscr();
 fp1=fopen("macroin.dat","r");
 fp2=fopen("macroout.dat","w");
 n=0;
 rewind(fp1);
 fscanf(fp1,"%s%s%s",ilab,iopd,oper);
 while(!feof(fp1))
 {
  if(strcmp(iopd,"MACRO")==0)
  {
   strcpy(NAMTAB[n],ilab);
   DEFTAB=fopen(NAMTAB[n],"w");
   fscanf(fp1,"%s%s%s",ilab,iopd,oper);
   while(strcmp(iopd,"MEND")!=0)
   {
    fprintf(DEFTAB,"%s\t%s\t%s\n",ilab,iopd,oper);
    fscanf(fp1,"%s%s%s",ilab,iopd,oper);
   }
   fclose(DEFTAB);
   n++;
  }
  else
  {
   flag=0;
   for(i=0;i<n;i++)
   {
    if(strcmp(iopd,NAMTAB[i])==0)
    {
     flag=1;
     DEFTAB=fopen(NAMTAB[i],"r");
     fscanf(DEFTAB,"%s%s%s\n",ilab,iopd,oper);
     while(!feof(DEFTAB))
     {
      fprintf(fp2,"%s\t%s\t%s\n",ilab,iopd,oper);
      fscanf(DEFTAB,"%s%s%s",ilab,iopd,oper);
     }
     break;
    }
   }
   if(flag==0)
   fprintf(fp2,"%s\t%s\t%s\n",ilab,iopd,oper);
  }
  fscanf(fp1,"%s%s%s",ilab,iopd,oper);
 }
 fprintf(fp2,"%s\t%s\t%s\n",ilab,iopd,oper);
 getch();
}
```

INPUT:



```
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 **
```

OUTPUT:



```
**      START   1000
**      LDA     N1
**      MUL     N2
**      STA     N5
**      LDA     N1
**      SUB     N2
**      STA     N4
**      LDA     N1
**      ADD     N2
**      STA     N3
**      END     **
**      END     **
```

RESULT:

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

Department of Computer

EngineeringAcademic

Term: Jan-Apr 2022

Class : T.E Computer Sem -VII
Subject: System Programming and Compiler Construction

Practical No:	**4**
Title:	Design Lexical Analyzer using High Level Language.
Date of Performance:	11/04/2022
Date of Submission:	11/04/2022
Roll No:	**8940**
Name of the Student:	**Warren Fernandes**

Evaluation:

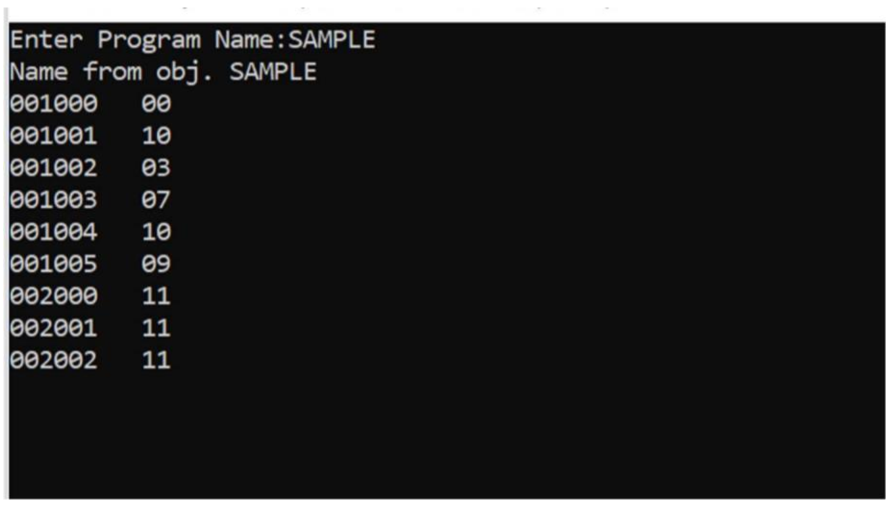
Sr. No	**Rubric**	**Grade**
1	**On time Completion & Submission (2)**	
2	**Output (3)**	
3	**Code Optimization (3)**	
4	**Knowledge of the topic (2)**	
5	**Total (10)**	

Signature of the Teacher:

Warren Fernandes TE COMPS B (8940) BATCH B

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
    void main()
    {
      FILE *fp;       int
i,addr1,l,j,staddr1;
      char name[10],line[50],name1[10],addr[10],rec[10],ch,staddr[10];

      printf("Enter Program Name:" );
scanf("%s",name);

      fp=fopen("input.txt","r");
      fscanf(fp,"%s",line);

      for(i=2,j=0;i<8,j<6;i++,j++)
name1[j]=line[i];
        name1[j]='\0';
      printf("Name from obj. %s\n",name1);

      if(strcmp(name,name1)==0)
       {
do
        {
                        fscanf(fp,"%s",line);

if(line[0]=='T')
                        {
                        for(i=2,j=0;i<8,j<6;i++,j++)
staddr[j]=line[i];                    staddr[j]='\0';
staddr1=atoi(staddr);
                        i=12;

                        while(line[i]!='$')
                        {
                          if(line[i]!='^')
                          {
                            printf("00%d \t %c%c\n", staddr1,line[i],line[i+1]);
                             staddr1++;
    i=i+2;
                          }
                          else i++;
                        }
                        }
                                    else
if(line[0]='E')
fclose(fp);
        }while(!feof(fp));
        }

    }
```

OUTPUT



```
Enter Program Name:SAMPLE
Name from obj. SAMPLE
001000   00
001001   10
001002   03
001003   07
001004   10
001005   09
002000   11
002001   11
002002   11
```

Department of Computer Engineering

Academic Term: Jan-Apr 2022

Class : T.E Computer Sem -VII
Subject: System Programming and Compiler Construction

Practical No:	**5**
Title:	**IMPLEMENTATION OF LEXICAL ANALYZER USING LEX TOOL**
Date of Performance:	11/04/2022
Date of Submission:	11/04/2022
Roll No:	**8940**
Name of the Student:	**Warren Fernandes**

Evaluation:

Sr. No	**Rubric**	**Grade**
1	**On time Completion & Submission (2)**	
2	**Output (3)**	
3	**Code Optimization (3)**	
4	**Knowledge of the topic (2)**	
5	**Total (10)**	

Signature of the Teacher:

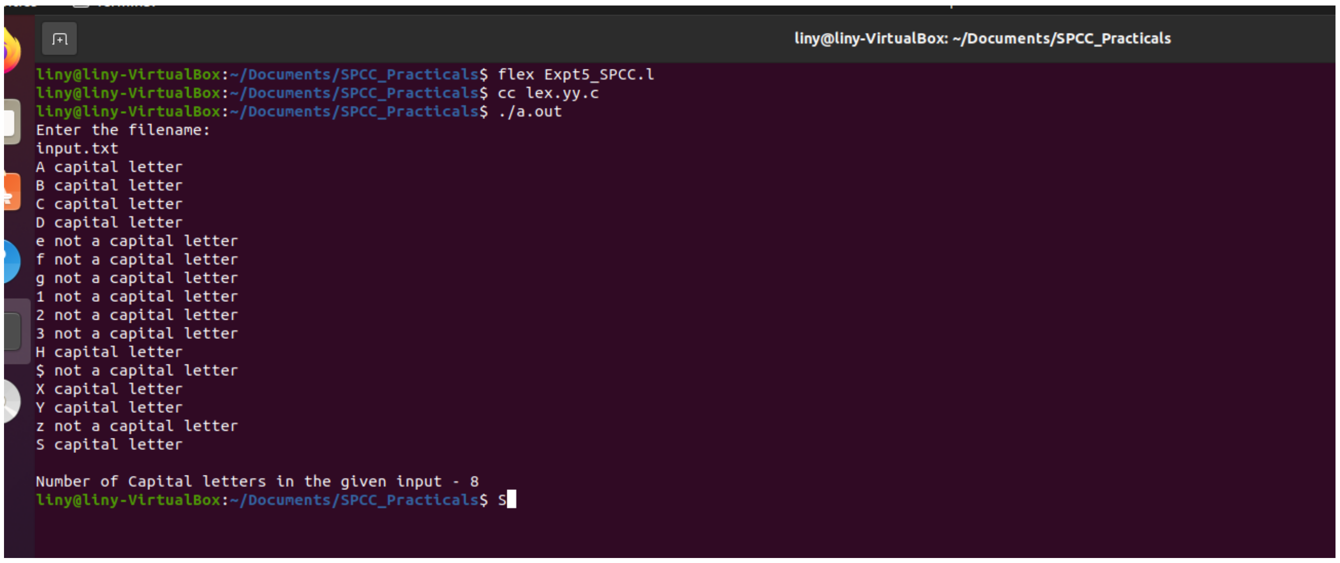
CODE:

```
/*** Definition Section has one variable which can be accessed inside yylex() and main() ***/
%{
int count = 0;
%}

/*** Rule Section has three rules, first rule matches with capital letters, second rule matches with any
character except newline and third rule does not take input after the enter***/
%%
[A-Z] {printf("%s capital letter\n", yytext);
   count++;}
.  {printf("%s not a capital letter\n", yytext);}
\n {return 0;}
%%

/*** Code Section prints the number of capital letter present in the given input***/
int yywrap(){}
int main (){

// Explanation:
// yywrap() - wraps the above rule section
/* yyin - takes the file pointer which contains the input*/
/* yylex() - this is the main flex function which runs the Rule Section*/
// yytext is the text in the buffer

// Uncomment the lines below
// to take input from file
// FILE *fp;
// char filename[50];
// printf("Enter the filename: \n");
// scanf("%s",filename);
// fp = fopen(filename,"r");
// yyin = fp;

yylex();
printf("\nNumber of Capital letters " "in the given input - %d\n", count);
return 0;
}
```

INPUT:

ABCDefg123H$XYzS

OUTPUT:



liny@liny-VirtualBox: ~/Documents/SPCC_Practicals

```
liny@liny-VirtualBox:~/Documents/SPCC_Practicals$ flex Expt5_SPCC.l
liny@liny-VirtualBox:~/Documents/SPCC_Practicals$ cc lex.yy.c
liny@liny-VirtualBox:~/Documents/SPCC_Practicals$ ./a.out
Enter the filename:
input.txt
A capital letter
B capital letter
C capital letter
D capital letter
e not a capital letter
f not a capital letter
g not a capital letter
1 not a capital letter
2 not a capital letter
3 not a capital letter
H capital letter
$ not a capital letter
X capital letter
Y capital letter
z not a capital letter
S capital letter

Number of Capital letters in the given input - 8
liny@liny-VirtualBox:~/Documents/SPCC_Practicals$ S
```

Department of Computer

EngineeringAcademic

Term: Jan-Apr 2022

Class : T.E Computer Sem -VII
Subject: System Programming and Compiler Construction

Practical No:	**6**
Title:	**Design of top Down Parser- Recursive Descent Parser.**
Date of Performance:	11/04/2022
Date of Submission:	11/04/2022
Roll No:	**8940**
Name of the Student:	**Warren Fernandes**

Evaluation:

Sr. No	**Rubric**	**Grade**
1	**On time Completion & Submission (2)**	
2	**Output (3)**	
3	**Code Optimization (3)**	
4	**Knowledge of the topic (2)**	
5	**Total (10)**	

Signature of the Teacher:

CODE

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

char input[100];

int i,l;

void main()

{

printf("\nRecursive descent parsing for the following grammar\n"); printf("\nE->TE'\nE'->+TE'/@\nT->FT'\nT'->*FT'/@\nF->(E)/ID\n"); printf("\nEnter the string to be checked:"); gets(input);

if(E())

{

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

printf("\nString is accepted");

else

printf("\nString is not accepted");

}

else

printf("\nString not accepted");

getch();
```

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

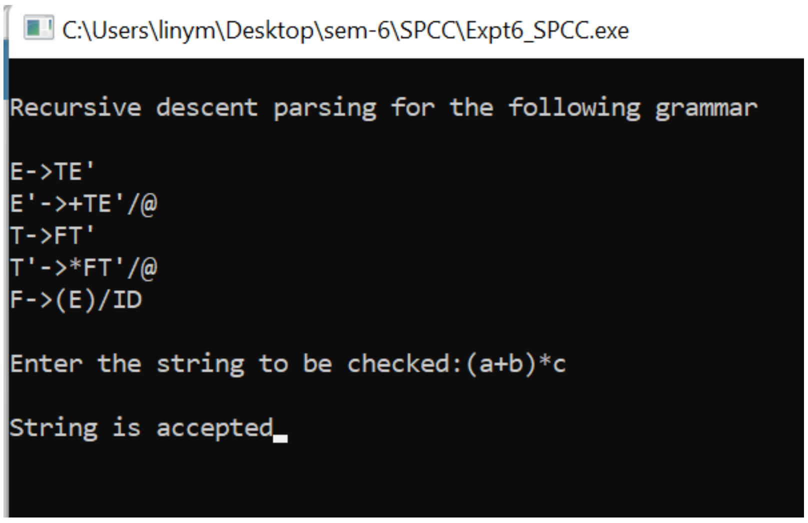
```
if(EP())

return(1);

else

return(0);

}

else

return(0);

}

else

return(1);

}

T()

{

if(F())

{

if(TP())

return(1);

else

return(0);

}
```

```
else

return(0);

}

TP()

{

if(input[i]=='*')

{

i++;

if(F())

{

if(TP())

return(1);

else

return(0);

}

else

return(0);

}

else

return(1);
```

```
}

F()

{

if(input[i]=='(')

{

i++;

if(E())

{

if(input[i]==')')

{

i++;

return(1);

}

else

return(0);

}

else

return(0);

}

else if(input[i]>='a'&&input[i]<='z'||input[i]>='A'&&input[i]<='Z')
```

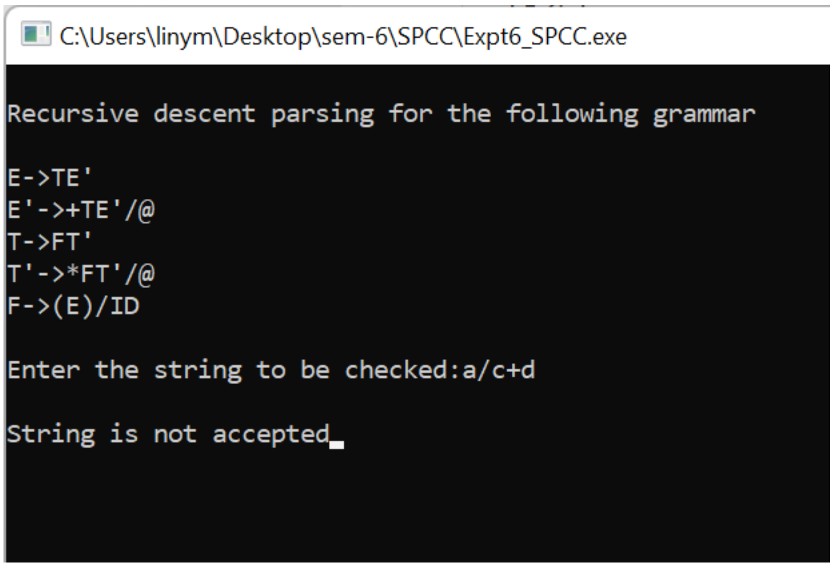
```
{
i++;
return(1);
}
else
return(0);
}
```

OUTPUT



C:\Users\linym\Desktop\sem-6\SPCC\Expt6_SPCC.exe

```
Recursive descent parsing for the following grammar

E->TE'
E'->+TE'/@
T->FT'
T'->*FT'/@
F->(E)/ID

Enter the string to be checked:(a+b)*c

String is accepted
```



C:\Users\linym\Desktop\sem-6\SPCC\Expt6_SPCC.exe

```
Recursive descent parsing for the following grammar

E->TE'
E'->+TE'/@
T->FT'
T'->*FT'/@
F->(E)/ID

Enter the string to be checked:a/c+d

String is not accepted
```

Department of Computer

EngineeringAcademic

Term: Jan-Apr 2022

Class : T.E Computer Sem -VII
Subject: System Programming and Compiler Construction

Practical No:	**7**
Title:	**Intermediate Code Generation OR Code Generation Phase**
Date of Performance:	11/04/2022
Date of Submission:	11/04/2022
Roll No:	**8940**
Name of the Student:	**Warren Fernandes**

Evaluation:

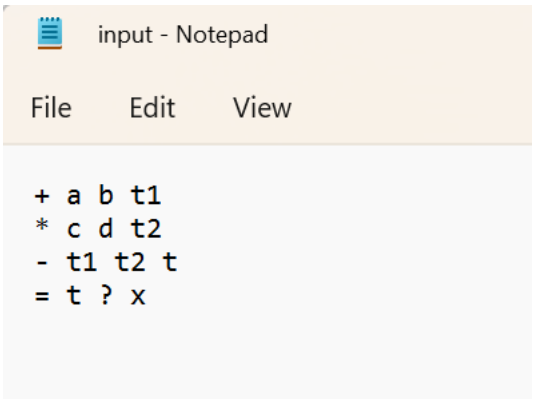
Sr. No	**Rubric**	**Grade**
1	**On time Completion & Submission (2)**	
2	**Output (3)**	
3	**Code Optimization (3)**	
4	**Knowledge of the topic (2)**	
5	**Total (10)**	

Signature of the Teacher:

CODE

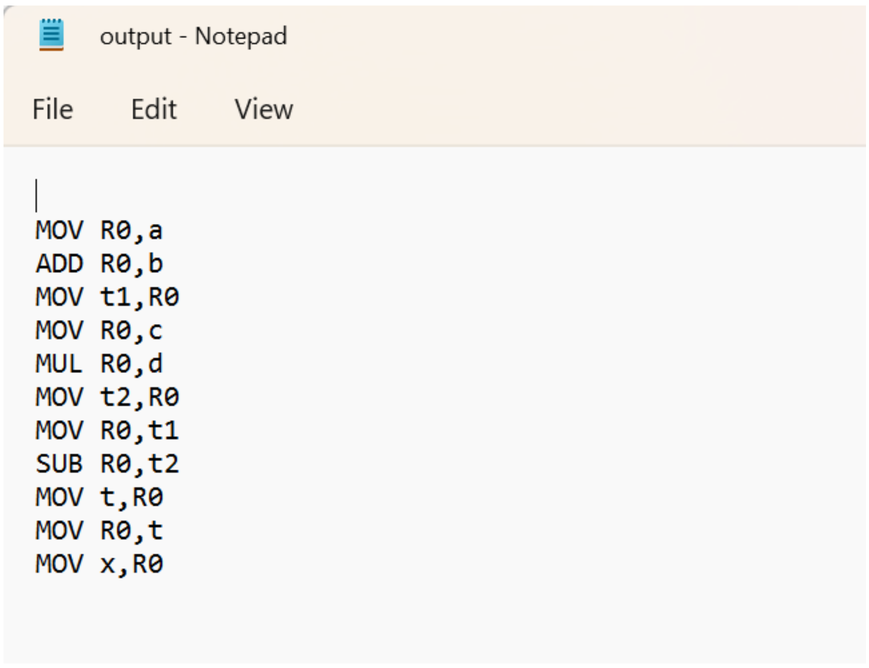
```
#include<stdio.h>
#include<conio.h>
#include<string.h>
char op[2],arg1[5],arg2[5],result[5];
void main()
{
  FILE *fp1,*fp2;
  fp1=fopen("input.txt","r");
  fp2=fopen("output.txt","w");
  while(!feof(fp1))
  {

    fscanf(fp1,"%s%s%s%s",op,arg1,arg2,result);
    if(strcmp(op,"+")==0)
    {
      fprintf(fp2,"\nMOV R0,%s",arg1);
      fprintf(fp2,"\nADD R0,%s",arg2);
      fprintf(fp2,"\nMOV %s,R0",result);
    }
     if(strcmp(op,"*")==0)
    {
      fprintf(fp2,"\nMOV R0,%s",arg1);
      fprintf(fp2,"\nMUL R0,%s",arg2);
      fprintf(fp2,"\nMOV %s,R0",result);
    }
    if(strcmp(op,"-")==0)
    {
      fprintf(fp2,"\nMOV R0,%s",arg1);
      fprintf(fp2,"\nSUB R0,%s",arg2);
      fprintf(fp2,"\nMOV %s,R0",result);
    }
      if(strcmp(op,"/")==0)
    {
      fprintf(fp2,"\nMOV R0,%s",arg1);
      fprintf(fp2,"\nDIV R0,%s",arg2);
      fprintf(fp2,"\nMOV %s,R0",result);
    }
if(strcmp(op,"=")==0)
    {
      fprintf(fp2,"\nMOV R0,%s",arg1);
      fprintf(fp2,"\nMOV %s,R0",result);
    }
    }
    fclose(fp1);
    fclose(fp2);
    getch();
  }
```

INPUT



```
+ a b t1
* c d t2
- t1 t2 t
= t ? x
```

OUTPUT



```
MOV R0,a
ADD R0,b
MOV t1,R0
MOV R0,c
MUL R0,d
MOV t2,R0
MOV R0,t1
SUB R0,t2
MOV t,R0
MOV R0,t
MOV x,R0
```