Unit 9
Lab

Lab no 1: Write program in C to test whether given entered string within valid comment section or not.

```
#include<stdio.h>
#include<conio.h>
int main()
{
        char com[30];
        int i=2, a=0;
        printf("\n Enter comment:");
        gets(com);
        if(com[0]=='/')
        {
                if(com[1]=='/')
                        printf("\n It is a comment");
                else if(com[1]=='*')
                {
                        for(i=2;i<=30;i++)
                        {
                                if(com[i]=='*'&&com[i+1]=='/')
                                {
                                        printf("\n It is a comment");
                                                a=1;
                                                break;
                                }
                                else
                                        continue;
                        }
                        if(a==0)
                                printf("\n It is not a comment");
                }
                else
                        printf("\n It is not a comment");
        }
        else
                printf("\n It is not a comment");
        return 0;
 }
```

Input/output

Run 1:

```
Enter comment: Hello
It is not a comment
```
Run 2:
```
Enter comment: /*New summit College*/
It is a comment
```
Run 3:
```
Enter comment: //This is a comment section
It is a comment
```

Lab no 2: Write a C program to recognize strings under 'a*', 'a*b+', 'abb'
```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int main()
{
	char s[20], c;
	int state=0, i=0;
	printf("\n Enter a string:");
	gets(s);
	while(s[i]!='\0')
	{
		switch(state)
		{
			case 0:
				c=s[i++];
				if(c=='a')
					state=1;
					else if(c=='b')
					state=2;
				else
					state=6;
					break;
			case 1:
				c=s[i++];
				if(c=='a')
					state=3;
					else if(c=='b')
					state=4;
				else
					state=6;
				break;
			case 2:
```

```
                    c=s[i++];
                    if(c=='a')
                              state=6;
                    else if(c=='b')
                              state=2;
                    else
                              state=6;
                    break;
          case 3:
                    c=s[i++];
                    if(c=='a')
                              state=3;
                     else if(c=='b')
                              state=2;
                     else
                              state=6;
                    break;
          case 4:
                    c=s[i++];
                     if(c=='a')
                              state=6;
                    else if(c=='b')
                              state=5;
                    else
                              state=6;
                     break;
          case 5:
                    c=s[i++];
                    if(c=='a')
                              state=6;
                    else if(c=='b')
                              state=2;
                    else
                              state=6;
                    break;
          case 6:
                    printf("\n %s is not recognized" ,s);
                    exit(0);
     }
}
if(state==1)
          printf("\n %s is accepted under rule 'a'", s);
else if((state==2)||(state==4))
```

```
        printf("\n %s is accepted under rule 'a*b+'", s);
    else if(state==5)
        printf("\n %s is accepted under rule 'abb'", s);
    return 0;
}
```

Input/output

Run 1:

```
Enter a string: aaaaabbbb

aaaaabbbb is accepted under rule 'a*b+'
```

Run 2:

```
Enter a string: bbbbaaaab
bbbbaaaab is not recognized
```

Run 3:

```
Enter a string: abb
abb is accepted under rule 'abb'
```

Lab no 3: Write a C program to test whether a given identifier is valid or not

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
int main()
{
    char a[10];
    int flag, i=1;
    printf("\n Enter an identifier:");
    gets(a);
    if(isalpha(a[0]) || a[0]=='_')
        flag=1;
    else
        printf("\n Not a valid identifier");
    while(a[i]!='\0')
    {
        if(!isdigit(a[i]) && !isalpha(a[i]) &&a[i]! = '_')
        {
            flag=0;
            break;
        }
        i++;
    }
```

```
	if(flag==1)
		printf("\n Valid identifier");
	else
		printf("Not a valid identifier");
	return 0;
}
```
Input/output
Run 1:

```
Enter an identifier: area_12no
Valid identifier
```
Run 2:

```
Enter an identifier: _sum5
Valid identifier
```
Run 3:

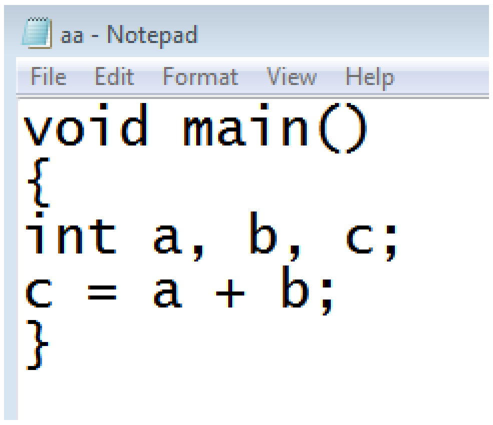
```
Enter an identifier: var@num
Not a valid identifier
```

Lab no 4: Program for Lexical Analyzer in C

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
 int isKeyword(char buffer[]){
	char keywords[32][10] = {"auto","break","case","char","const","continue","default",

	"do","double","else","enum","extern","float","for","goto",
		"if","int","long","register","return","short","signed",
			"sizeof","static","struct","switch","typedef","union",
					"unsigned","void","volatile","while"};
	int i, flag = 0;
	for(i = 0; i < 32; ++i)
	{
		if(strcmp(keywords[i], buffer) == 0)
		{
			flag = 1;
			break;
		}
	}
	return flag;
}

int main()
```

```
{
        char ch, buffer[15], operators[] = "+-*/%=";
        FILE *fp;
        int i,j=0;
        fp = fopen("aa.txt","r");
        if(fp == NULL)
        {
                printf("error while opening the file\n");
                exit(0);
        }
        while((ch = fgetc(fp)) != EOF)
        {
                for(i = 0; i < 6; ++i)
                {
                        if(ch == operators[i])
                                printf("%c is operator\n", ch);
                }
                if(isalnum(ch))
                {
                        buffer[j++] = ch;
                }
                else if((ch == ' ' || ch == '\n') && (j != 0))
                {
                                buffer[j] = '\0';
                                j = 0;

                                if(isKeyword(buffer) == 1)
                                        printf("%s is keyword\n", buffer);
                                else
                                        printf("%s is identifier\n", buffer);
                }
        }
        fclose(fp);
        return 0;
}
```

Input File format is:



```c
void main()
{
int a, b, c;
c = a + b;
}
```

Output

```
void is keyword
main is identifier
int is keyword
a is identifier
b is identifier
c is identifier
c is identifier
= is operator
a is identifier
+ is operator
b is identifier
```

Lab no 5: C- program to implement first of a given grammar

```c
#include<stdio.h>
#include<ctype.h>
void FIRST(char[ ], char );
void addToResultSet(char[ ], char);
int numOfProductions;
char productionSet[10][10];
int main()
{
   int i;
   char choice;
   char c;
   char result[20];
   printf("How many number of productions ? :");
```

```
    scanf(" %d", &numOfProductions);
    for(i=0; i <numOfProductions; i++)//read production string e.g.: E=E+T
    {
        printf("Enter productions Number %d : ",i+1);
        scanf(" %s", productionSet[i]);
    }
    do
    {
        printf("\n Find the FIRST of  :");
        scanf(" %c", &c);
        FIRST(result, c); //Compute FIRST; Get Answer in 'result' array
        printf("\n FIRST(%c)= { ",c);
        for(i=0;result[i]!='\0';i++)
        printf(" %c ",result[i]);       //Display result
        printf("}\n");
         printf("press 'y' to continue : ");
        scanf(" %c", &choice);
    }while(choice=='y'||choice =='Y');
}
void FIRST(char* Result, char c)
{
    int i, j, k;
    char subResult[20];
    int foundEpsilon;
    subResult[0]='\0';
    Result[0]='\0';
    //If X is terminal, FIRST(X) = {X}
    if(!(isupper(c)))
    {
            addToResultSet(Result, c);
            return ;
    }
    //If X is non terminal then read each production
    for(i=0; i<numOfProductions; i++)
```

```
  {
        //Find production with X as LHS
        if(productionSet[i][0]==c)
        {
                if(productionSet[i][2]=='$')
                        addToResultSet(Result,'$');
                //If X is a non-terminal, and X → Y1 Y2 … Yk is a production, then add a
to FIRST(X)
                else
                        {
                        j=2;
                        while(productionSet[i][j]!='\0')
                        {
                                foundEpsilon=0;
                                FIRST(subResult, productionSet[i][j]);
                                for(k=0;subResult[k]!='\0';k++)
                                        addToResultSet(Result, subResult[k]);
                                for(k=0;subResult[k]!='\0';k++)
                                {
                                        if(subResult[k]=='$')
                                        {
                                                        foundEpsilon=1;
                                                        break;
                                        }
                                }
           //No e found, no need to check next element
           if(!foundEpsilon)
                break;
           j++;
          }
        }
    }
 }
 return;
```

```
}
void addToResultSet(char Result[ ], char val)
{
   int k;
   for(k=0 ;Result[k]!='\0';k++)
      if(Result[k]==val)
         return;
   Result[k]=val;
   Result[k+1]='\0';
}
```
Input/output

```
How many numbers of productions? : 5
Enter productions Number 1: S=L=R
Enter productions Number 2: S=R
Enter productions Number 3: L=*R
Enter productions Number 4: L=a
Enter productions Number 5: R=L
Find the FIRST of: S
         FIRST(S) = { *  a }
Press 'y' to continue: y
Find the FIRST of: L
         FIRST(L)= {  *  a }
Press 'y' to continue:
Find the FIRST of: a
         FIRST(a)= { a }
Press 'y' to continue: y
Find the FIRST of: *R
         FIRST(*R)= {  *  }
Press 'y' to continue:
```

Lab no 6: C-Program to Calculate Follow(A)

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
int n,p,i=0,j=0;
char a[10][10],Result[10];
char subResult[20];
```

```
void follow(char* Result,char c);
void first(char* Result,char c);
void addToResultSet(char[ ], char);
int main()
{
        int i;
        int choice;
        char c, ch;
        printf("Enter the no. of productions: ");
        scanf("%d", &n);
        printf(" Enter %d productions\n Production with multiple terms should be give
        as separate productions \n", n);
        for(i=0;i<n;i++)
         scanf("%s", a[i]);
        do
         {
                printf("Find FOLLOW of -->");
                scanf(" %c", &c);
                follow(Result, c);
                printf("FOLLOW(%c) = { ", c);
                for(i=0;Result[i]!='\0';i++)
                        printf(" %c ", Result[i]);
                printf(" }\n");
                printf("Do you want to continue(Press 1 to continue....)?");
                scanf("%d", &choice);
                }while(choice==1);
        }
        void follow(char* Result, char c)
        {
                int k;
                subResult[0]='\0';
                Result[0]='\0';
                if(a[0][0]==c) addToResultSet(Result,'$');
                for(i=0;i<n;i++)
                {
                        for(j=2;j<strlen(a[i]);j++)
                        {
                                if(a[i][j]==c)
                                {
                                        if(a[i][j+1]!='\0')first(subResult,a[i][j+1]);
                                                if(a[i][j+1]=='\0'&&c!=a[i][0])
                                                        follow(subResult,a[i][0]);
                                        for(k=0;subResult[k]!='\0';k++)
                                        addToResultSet(Result,subResult[k]);
                              }
                         }
                    }
          }
          void first(char* R, char c)
          {
                    int k, m;
                    if(!(isupper(c))&&c!='#')
                              addToResultSet(R, c);
                     for(k=0;k<n;k++)
                     {
                              if(a[k][0]==c)
                              {
                                        if(a[k][2]=='#'&&c!=a[i][0])
                                                  follow(R, a[i][0]);
                                         else if((!(isupper(a[k][2])))&&a[k][2]!='#')
                                            addToResultSet(R, a[k][2]);
                                         else first(R, a[k][2]);
                                                  for(m=0;R[m]!='\0';m++)
                                                            addToResultSet(Result, R[m]);
                              }
                    }
          }
          void addToResultSet(char Result[], char val)
          {
                    int k;
                    for(k=0 ;Result[k]!='\0';k++)
                               if(Result[k]==val)
                                        return;
                              Result[k]=val;
                              Result[k+1]='\0';
          }
```
Input/output

```
Enter the no. of productions: 5
Enter 5 productions
Production with multiple terms should be give as separate productions
R=aS
R=(R)S
S=+RS
S=aRS
```

```
S=*S
Find FOLLOW of -->R
FOLLOW(R) = { $  )  +  a  *  }
Do you want to continue (Press 1 to continue....)? 1
Find FOLLOW of -->S
FOLLOW(S) = { $  )  +  a  *  }
Do you want to continue (Press 1 to continue....)?
```

Lab no 7: Write a C program for constructing of LL (1) parsing

```
#include<stdio.h>
#include<string.h>
#include<process.h>
char s[20],stack[20];
int main()
 {
        char  m[5][6][4]={"tb"," "," ","tb"," "," "," ","+tb"," "," ","n","n","fc"," "," ","fc"," "," ","
        ","n","*fc"," a","n","n","i"," "," ","(e)"," "," "};
        int size[5][6]={2,0,0,2,0,0,0,3,0,0,1,1,2,0,0,2,0,0,0,1,3,0,1,1,1,0,0,3,0,0};
        int i,j,k,n,str1,str2;
        printf("\n Enter the input string: ");
        scanf("%s",s);
        strcat(s,"$");
        n=strlen(s);
        stack[0]='$';
        stack[1]='e';
        i=1;
        j=0;
        printf("\nStack     Input\n");
        printf("__________________\n");
        while((stack[i]!='$')&&(s[j]!='$'))
        {
                if(stack[i]==s[j])
                {
                        i--;
                        j++;
                }
                switch(stack[i])
                {
                        case 'e': str1=0;
                        break;
                        case 'b': str1=1;
```

```
            break;
            case 't': str1=2;
            break;
            case 'c': str1=3;
            break;
            case 'f': str1=4;
            break;
}
switch(s[j])
{
             case 'i': str2=0;
            break;
            case '+': str2=1;
            break;
             case '*': str2=2;
            break;
            case '(': str2=3;
            break;
             case ')': str2=4;
            break;
             case '$': str2=5;
            break;
 }
if(m[str1][str2][0]=='\0')
 {
             printf("\nERROR");
            exit(0);
   }
 else if(m[str1][str2][0]=='n')
            i--;
else if(m[str1][str2][0]=='i')
            stack[i]='i';
else
{
            for(k=size[str1][str2]-1;k>=0;k--)
            {
                      stack[i]=m[str1][str2][k];
                      i++;
            }
            i--;
}
for(k=0;k<=i;k++)
            printf(" %c",stack[k]);
```

```
              printf("       ");
          for(k=j;k<=n;k++)
              printf("%c",s[k]);
          printf(" \n ");
   }
   printf("\n SUCCESS");
   return 0;
}
```

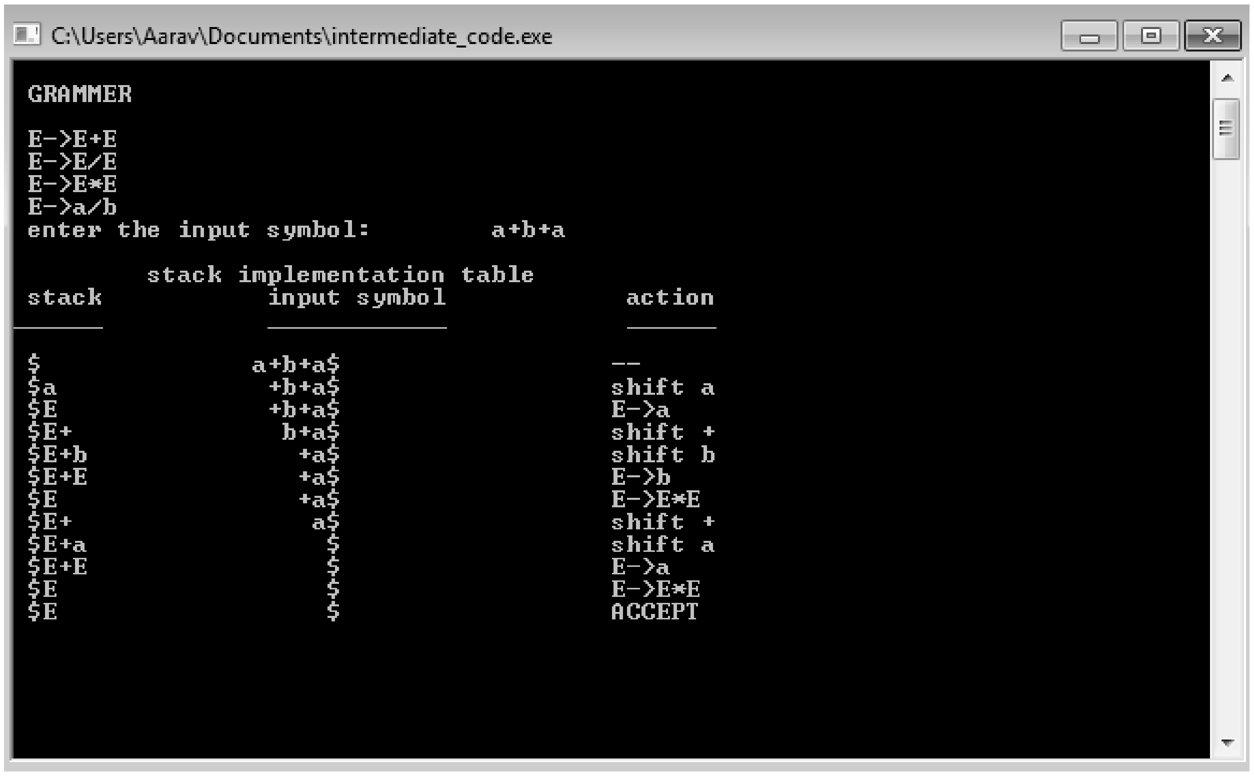
Input/output

```
Enter the input string: i*i+i
Stack          Input
_____________________
 $ b t          i*i+i$
  $ b c f       i*i+i$
  $ b c i       i*i+i$
  $ b c f *     *i+i$
  $ b c i       i+i$
  $ b                  +i$
  $ b t +       +i$
  $ b c f       i$
  $ b c i       i$
  $ b                  $
 SUCCESS
--------------------------------
```

Lab no 8: C Program to Implement Shift Reduce Parser

```
#include <stdio.h>
#include<stdlib.h>
#include<conio.h>
#includ<string.h>
char ip_sym[15],stack[15];
int ip_ptr=0,st_ptr=0,len,i;
char temp[2],temp2[2];
char act[15];
void check();
void main()
{
          clrscr();
          printf("\n\t\t SHIFT REDUCE PARSER\n");
          printf("\n GRAMMER\n");
          printf("\n E->E+E\n E->E/E");
          printf("\n E->E*E\n E->a/b");
          printf("\n enter the input symbol:\t");
```

```
        gets(ip_sym);
        printf("\n\t stack implementation table");
        printf("\n stack\t\t input symbol\t\t action");
        printf("\n______\t\t ____________\t\t ______\n");
        printf("\n $\t\t%s$\t\t\t--",ip_sym);
        strcpy(act,"shift ");
        temp[0]=ip_sym[ip_ptr];
        temp[1]='\0';
        strcat(act,temp);
        len=strlen(ip_sym);
        for(i=0;i<=len-1;i++)
        {
                stack[st_ptr]=ip_sym[ip_ptr];
                stack[st_ptr+1]='\0';
                ip_sym[ip_ptr]=' ';
                ip_ptr++;
                printf("\n $%s\t\t%s$\t\t\t%s",stack,ip_sym,act);
                strcpy(act,"shift ");
                temp[0]=ip_sym[ip_ptr];
                temp[1]='\0';
                strcat(act,temp);
                check();
                st_ptr++;
        }
        st_ptr++;
        check();
}
void check()
{
        int flag=0;
        temp2[0]=stack[st_ptr];
        temp2[1]='\0';
        if((!strcmpi(temp2,"a"))||(!strcmpi(temp2,"b")))
        {
                stack[st_ptr]='E';
                if(!strcmpi(temp2,"a"))
                        printf("\n $%s\t\t%s$\t\t\tE->a",stack, ip_sym);
                else
                        printf("\n $%s\t\t%s$\t\t\tE->b",stack,ip_sym);
                flag=1;
        }
        if((!strcmpi(temp2,"+"))||(strcmpi(temp2,"*"))||(!strcmpi(temp2,"/")))
        {
                flag=1;
        }
        if((!strcmpi(stack,"E+E"))||(!strcmpi(stack,"E\E"))||(!strcmpi(stack,"E*E")))
        {
                strcpy(stack, "E");
                st_ptr=0;
                if(!strcmpi(stack,"E+E"))
                        printf("\n $%s\t\t%s$\t\t\tE->E+E", stack,  ip_sym);
                else
                        if(!strcmpi(stack,"E\E"))
                                printf("\n $%s\t\t %s$\t\t\tE->E\E",stack,ip_sym);
                        else
                                printf("\n $%s\t\t%s$\t\t\tE->E*E",stack,ip_sym);
                        flag=1;
        }
        if(!strcmpi(stack,"E")&&ip_ptr==len)
        {
                printf("\n $%s\t\t%s$\t\t\tACCEPT",stack,ip_sym);
                getch();
                exit(0);
        }
        if(flag==0)
        {
                printf("\n%s\t\t\t%s\t\t reject",stack,ip_sym);
                exit(0);
        }
        return;
}
```
Input/output



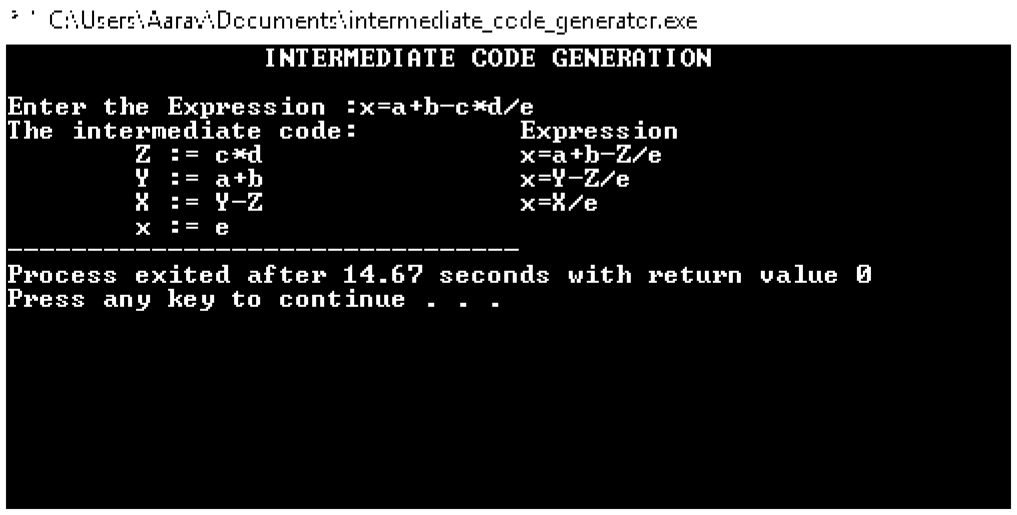
```
C:\Users\Aarav\Documents\intermediate_code.exe

GRAMMER

E->E+E
E->E/E
E->E*E
E->a/b
enter the input symbol:        a+b+a

        stack implementation table
stack            input symbol            action
_______          ____________            _______

$               a+b+a$                   --
$a               +b+a$                   shift a
$E               +b+a$                   E->a
$E+               b+a$                   shift +
$E+b               +a$                   shift b
$E+E               +a$                   E->b
$E                 +a$                   E->E*E
$E+                 a$                   shift +
$E+a                 $                   shift a
$E+E                 $                   E->a
$E                   $                   E->E*E
$E                   $                   ACCEPT
```

Lab no 9: C-program for intermediate Code Generation

```
#include<stdio.h>
#include<string.h>
#include<process.h>
int i=1,j=0,no=0,tmpch=90;
char str[100],left[15],right[15];
void findopr();
void explore();
void fleft(int);
void fright(int);
struct exp
{
        int pos;
        char op;
}k[15];
int main()
{
        printf("\t\t INTERMEDIATE CODE GENERATION\n\n");
```

```
            printf("Enter the Expression :");
            scanf("%s", str);
            printf("The intermediate code:\t\t Expression\n");
            findopr();
            explore();
            return 0;
}
void findopr()
{
            for(i=0;str[i]!='\0';i++)
             if(str[i]==':')
             {
                         k[j].pos=i;
                         k[j++].op=':';
             }
            for(i=0;str[i]!='\0';i++)
            if(str[i]=='/')
            {
                        k[j].pos=i;
                         k[j++].op='/';
            }
            for(i=0;str[i]!='\0';i++)
             if(str[i]=='*')
             {
                                    k[j].pos=i;
                                    k[j++].op='*';
              }

            for(i=0;str[i]!='\0';i++)
           if(str[i]=='+')
             {
                                    k[j].pos=i;
                                    k[j++].op='+';
              }
             for(i=0;str[i]!='\0';i++)
             {
                     if(str[i]=='-')
                     {
                             k[j].pos=i;
                             k[j++].op='-';
                     }
            }
}
void explore()
{
        i=1;
        while(k[i].op!='\0')
        {
                fleft(k[i].pos);
                fright(k[i].pos);
                str[k[i].pos]=tmpch--;
                printf("\t%c := %s%c%s\t\t", str[k[i].pos], left, k[i].op, right);
                for(j=0;j <strlen(str);j++)
                    if(str[j]!='$')
                printf("%c", str[j]);
                printf("\n");
                i++;
        }
        fright(-1);
        if(no==0)
        {
                fleft(strlen(str));
                printf("\t%s := %s", right, left);
                exit(0);
        }
        printf("\t%s :=  %c", right, str[k[--i].pos]);
}
void fleft(int x)
```

```
{
        int w=0, flag=0;
        x--;
        while(x!=  -1  &&str[x]!=  '+'  &&str[x]!='*'&&str[x]!='='&&str[x]!='\0'&&str[x]!='-
       '&& str[x]!='/' && str[x]!=':')
        {
                if(str[x]!='$'&& flag==0)
                {
                        left[w++]=str[x];
                        left[w]='\0';
                        str[x]='$';
                        flag=1;
                }
                x--;
        }
}
void fright(int x)
{
        int w=0,flag=0;
        x++;
        while(x!=          -1          &&          str[x]!=          '+'&&str[x]!='*'&&str[x]!='\0'&&
       str[x]!='='&&str[x]!=':'&& str[x]!='-'&& str[x]!='/')
        {
                if(str[x]!='$'&& flag==0)
                {
                        right[w++]=str[x];
                        right[w]='\0';
                        str[x]='$';
                        flag=1;
                }
               x++;
        }
}
```

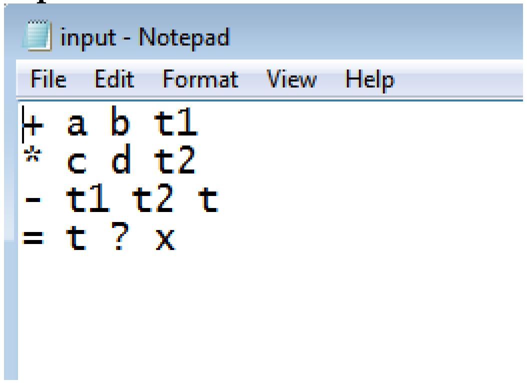


```
C:\Users\Aarav\Documents\intermediate_code_generator.exe
                 INTERMEDIATE CODE GENERATION

Enter the Expression :x=a+b-c*d/e
The intermediate code:          Expression
        Z := c*d                x=a+b-Z/e
        Y := a+b                x=Y-Z/e
        X := Y-Z                x=X/e
        x := e
--------------------------------
Process exited after 14.67 seconds with return value 0
Press any key to continue . . .
```

Lab no 9: C-program for Final Code Generation

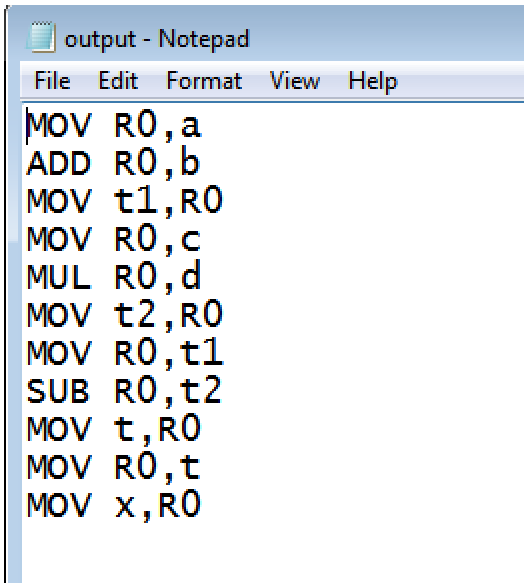
```
#include<stdio.h>
#include<string.h>
char op[2],arg1[5],arg2[5],result[5];
int main()
{
        FILE *fp1,*fp2;
        fp1=fopen("input.txt","r");
        fp2=fopen("output.txt","w");
        while(!feof(fp1))
        {
                fscanf(fp1,"%s%s%s%s",op,arg1,arg2,result);
                if(strcmp(op,"+")==0)
                {
                        fprintf(fp2,"\n MOV R0,%s",arg1);
                        fprintf(fp2,"\n ADD R0,%s",arg2);
                        fprintf(fp2,"\n MOV %s,R0",result);
                }
                if(strcmp(op,"*")==0)
                {
                        fprintf(fp2,"\n MOV R0,%s",arg1);
                        fprintf(fp2,"\n MUL R0,%s",arg2);
                        fprintf(fp2,"\n MOV %s, R0",result);
                }
                if(strcmp(op,"-")==0)
                {
                          fprintf(fp2,"\n MOV R0,%s",arg1);
                        fprintf(fp2,"\n SUB R0,%s",arg2);
                        fprintf(fp2,"\n MOV %s,R0",result);
                }
                if(strcmp(op,"/")==0)
                {
                        fprintf(fp2,"\n MOV R0,%s",arg1);
                        fprintf(fp2,"\n DIV R0,%s",arg2);
                        fprintf(fp2,"\n MOV %s,R0",result);
                }
                if(strcmp(op,"=")==0)
                {
                        fprintf(fp2,"\n MOV R0,%s",arg1);
                        fprintf(fp2,"\n MOV %s,R0",result);
                }
            }
            fclose(fp1);
            fclose(fp2);
            return 0;
           }
}
```

Input file



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

Output file



```
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
```