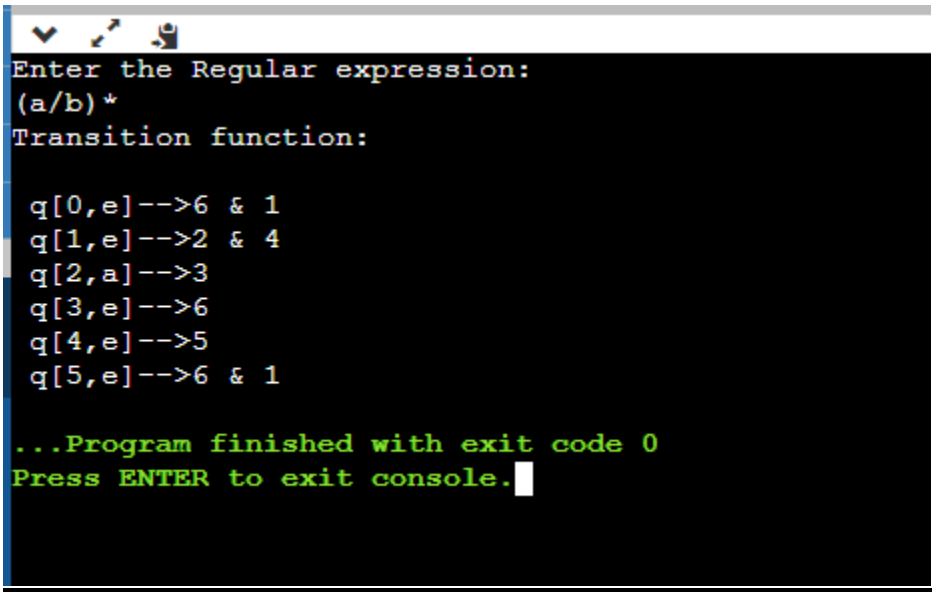
PROGRAM:

```
#include<stdio.h>
#include<string.h>
void main()
{
   char re[20];
   int q[20][3],i,j,len,a,b;
   for(a=0;a<20;a++)
   {
      for(b=0;b<3;b++)
      {
         q[a][b]=0;
      }
   }
   printf("Enter the Regular expression:\n");
   scanf("%s",re);
   len=strlen(re);
   i=0;j=1;
   while(i<len)
   {
      if(re[i]=='a'&&re[i+1]!='/'&&re[i+1]!='*')
      {
         q[j][0]=j+1;j++;
      }
      if(re[i]=='b'&&re[i+1]!='/'&&re[i+1]!='*')
      {
         q[j][1]=j+1;j++;
      }
      if(re[i]=='e'&&re[i+1]!='/'&&re[i+1]!='*')
      {
         q[j][2]=j+1;j++;
```

```
}
if(re[i]=='a'&& re[i+1]=='/'&&re[i+2]=='b')
{
   q[j][2]=((j+1)*10)+(j+3);j++;
   q[j][0]=j+1;j++;
   q[j][2]=j+3;j++;
   q[j][2]=j+1;j++;i=i+2;
}
if(re[i]=='b'&&re[i+1]=='/'&&re[i+2]=='a')
{
   q[j][2]=((j+1)*10)+(j+3);j++;
   q[j][1]=j+1;j++;
   q[j][2]=j+3;j++;
   q[j][0]=j+1;j++;
   q[j][2]=j+1;j++;
   i=i+2;
}
if(re[i]=='a'&&re[i+1]=='*')
{
   q[j][2]=((j+1)*10)+(j+3);
   j++;
   q[j][0]=j+1;
   j++;
   q[j][2]=((j+1)*10)+(j-1);
   j++;
}
if(re[i]=='b'&&re[i+1]=='*')
{
   q[j][2]=((j+1)*10)+(j+3);
   j++;
   q[i][j]=j+1;
```

```
            j++;
            q[j][2]=((j+1)*10)+(j-1);
            j++;
        }
        if(re[i]==')'&&re[i+1]=='*')
        {
            q[0][2]=((j+1)*10)+1;
            q[j][2]=((j+1)*10)+1;
            j++;
        }
        i++;
    }
    printf("Transition function:\n");
    for(i=0;i<=j;i++)
    {
        if(q[i][0]!=0)
            printf("\n q[%d,a]-->%d",i,q[i][0]);
        if(q[i][1]!=0)
            printf("\n q[%d,b]-->%d",i,q[i][1]);
        if(q[i][2]!=0)
        {
            if(q[i][2]<10)
                printf("\n q[%d,e]-->%d",i,q[i][2]);
            else
                printf("\n q[%d,e]-->%d & %d",i,q[i][2]/10,q[i][2]%10);
        }
    }
}
```

OUTPUT:

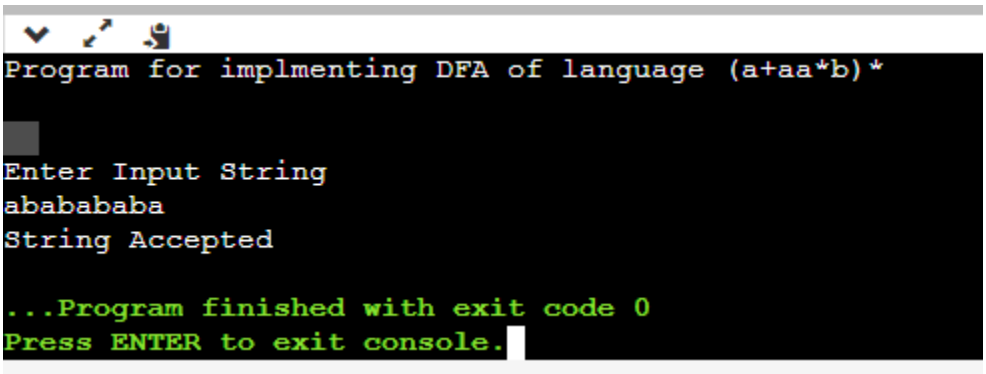

Enter the Regular expression:
(a/b)*
Transition function:
q[0,e]-->6 & 1
q[1,e]-->2 & 4
q[2,a]-->3
q[3,e]-->6
q[4,e]-->5
q[5,e]-->6 & 1
...Program finished with exit code 0
Press ENTER to exit console.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<strings.h>
void main()
{
int table[2][2],i,j,l,status=0,success;
char input[100];
printf("Program for implmenting DFA of language (a+aa*b)*\n\n\nEnter Input String \n");
table[0][0]=1;
table[0][1]=-1;
table[1][0]=1;
table[1][1]=0;
scanf("%s",input);
l=strlen(input);
for(i=0;i<l;i++)
{
        if(input[i]!='a'&&input[i]!='b')
        {
        printf("Value entered is wrong");
```

```
        getch();
        exit(0);
        }
 if(input[i]=='a')
status=table[status][0];
else
status=table[status][1];
if(status==-1)
{
        printf("String not Accepted by this Language");
        break;
}
}
if(i==l)
printf("String Accepted");
getch();
}
```

OUTPUT:


Program for implmenting DFA of language (a+aa*b)*
Enter Input String
abababababa
String Accepted
...Program finished with exit code 0
Press ENTER to exit console.

PROGRAM:

```
lex:
%{
int COMMENT=0;
%}
identifier [a-zA-Z][a-zA-Z0-9]*
%%
#.* { printf("\n%s is a PREPROCESSOR DIRECTIVE",yytext);}
int |
float |
char |
double |
while |
for |
do |
if |
break |
continue |
void |
switch |
case |
long |
struct |
const |
typedef |
return |
else |
goto {printf("\n\t%s is a KEYWORD",yytext);}
"/*" {COMMENT = 1;}
"*/" {COMMENT = 0;}
{identifier}\( {if(!COMMENT)printf("\n\nFUNCTION\n\t%s",yytext);}
```

```
\{ {if(!COMMENT) printf("\n BLOCK BEGINS");}
\} {if(!COMMENT) printf("\n BLOCK ENDS");}
{identifier}(\[[0-9]*\])? {if(!COMMENT) printf("\n %s IDENTIFIER",yytext);}
\".*\" {if(!COMMENT) printf("\n\t%s is a STRING",yytext);}
[0-9]+ {if(!COMMENT) printf("\n\t%s is a NUMBER",yytext);}
\)(\;)? {if(!COMMENT) printf("\n\t");ECHO;printf("\n");}
\( ECHO;
= {if(!COMMENT)printf("\n\t%s is an ASSIGNMENT OPERATOR",yytext);}
\<= |
\>= |
\< |
== |
\> {if(!COMMENT) printf("\n\t%s is a RELATIONAL OPERATOR",yytext);}
%%
int main(int argc,char **argv)
{
if (argc > 1)
{
FILE *file;
file = fopen(argv[1],"r");
if(!file)
{
printf("could not open %s \n",argv[1]);
exit(0);
}
yyin = file;
}
yylex();
printf("\n\n");
return 0;
} int yywrap()
```

```
{
return 0;
}
```

second.c

```
#include <stdio.h>
int main()
{
        int number;
        printf("Enter an integer: ");
        scanf("%d", &number);
        if(number % 2 == 0)
                printf("%d is even.", number);
        else
                printf("%d is odd.", number);
        return 0;
}
```

#include <stdio.h> is a PREPROCESSOR DIRECTIVE

int is a KEYWORD

FUNCTION

main(

)

BLOCK BEGINS

int is a KEYWORD

number IDENTIFIER;

FUNCTION
printf(
"Enter an integer: " is a STRING
);

FUNCTION
scanf(
"%d" is a STRING, &
number IDENTIFIER
);

FUNCTION
if(
number IDENTIFIER %
2 is a NUMBER
== is a RELATIONAL OPERATOR
0 is a NUMBER
)

FUNCTION
printf(
"%d is even." is a STRING,
number IDENTIFIER
);

else is a KEYWORD

FUNCTION

printf(

"%d is odd." is a STRING,

number IDENTIFIER

);

return is a KEYWORD

0 is a NUMBER;

BLOCK ENDS

OUTPUT:

C:\Users\18csec29\Desktop\pcd>flex first.l

C:\Users\18csec29\Desktop\pcd>gcc lex.yy.c

C:\Users\18csec29\Desktop\pcd>a.exe second.c

PROGRAM:

LEX PART:

```
%{
#include<stdio.h>
#include "y.tab.h"
extern int yylval;
%}
%%
[0-9]+ {
        yylval=atoi(yytext);
        return NUMBER;
      }
[\t] ;
[\n] return 0;
. return yytext[0];
%%
int yywrap()
{
return 1;
}
```

YACC PART:

```
%{

   #include<stdio.h>
   int flag=0;
%}
```

```
%token NUMBER
%left '+' '-'
%left '*' '/' '%'
%left '(' ')'
%%
ArithmeticExpression: E{
        printf("\nResult=%d\n",$$);
        return 0;
       };
E:E'+'E {$$=$1+$3;}
 |E'-'E {$$=$1-$3;}
 |E'*'E {$$=$1*$3;}
 |E'/'E {$$=$1/$3;}
 |E'%'E {$$=$1%$3;}
 |'('E')' {$$=$2;}
 | NUMBER {$$=$1;}
;
%%
void main()
{
   printf("\nEnter Any Arithmetic Expression which can have operations Addition, Subtraction,
Multiplication, Divison, Modulus and Round brackets:\n");
   yyparse();
  if(flag==0)
   printf("\nEntered arithmetic expression is Valid\n\n");
}
void yyerror()
{
   printf("\nEntered arithmetic expression is Invalid\n\n");
   flag=1;
}
```

OUTPUT:

C:\Users\18csec29\Desktop\Ex-4>yacc -d 4c.y

C:\Users\18csec29\Desktop\Ex-4>lex 4c.l

C:\Users\18csec29\Desktop\Ex-4>gcc lex.yy.c y. tab.c -w

C:\Users\18csec29\Desktop\Ex-4>./a.out

3*5+4

19.000000

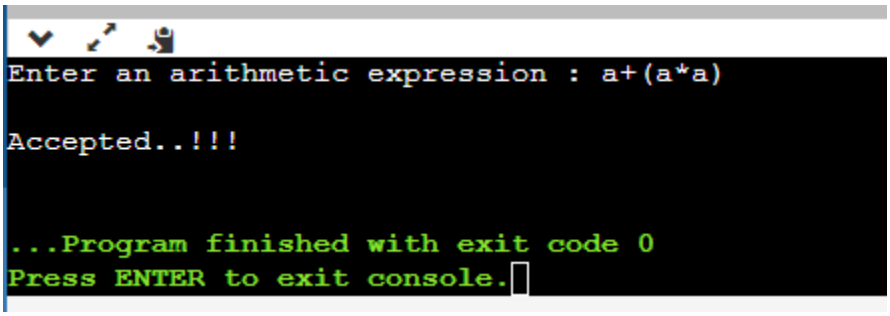
PROGRAM:

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
char input[10];
int i,error;
void E();
 void T();
void Eprime();
void Tprime();
void F();
main()
{
   i=0; error=0;
   printf("Enter an arithmetic expression : ");
   gets(input);
   E();
   if(strlen(input)==i&&error==0)
      printf("\nAccepted..!!!\n");
   else
      printf("\nRejected..!!!\n");
}
void E()
{
   T();
   Eprime();
}
void Eprime()
{
   if(input[i]=='+')
```

```
    {
        i++;
        T();
        Eprime();
     }
}
void T()
{
    F();
    Tprime();
}
void Tprime()
{
    if(input[i]=='*')
    {
        i++;
        F();
        Tprime();
    }
}
void F()
{
    if(isalnum(input[i]))
        i++;
    else if(input[i]=='(')
        {
            i++;
            E();
            if(input[i]==')')
                i++;
            else
```

```
            error=1;
        }
    else
        error=1;
}
```

OUTPUT:


Enter an arithmetic expression : a+(a*a)
Accepted..!!!
...Program finished with exit code 0
Press ENTER to exit console.

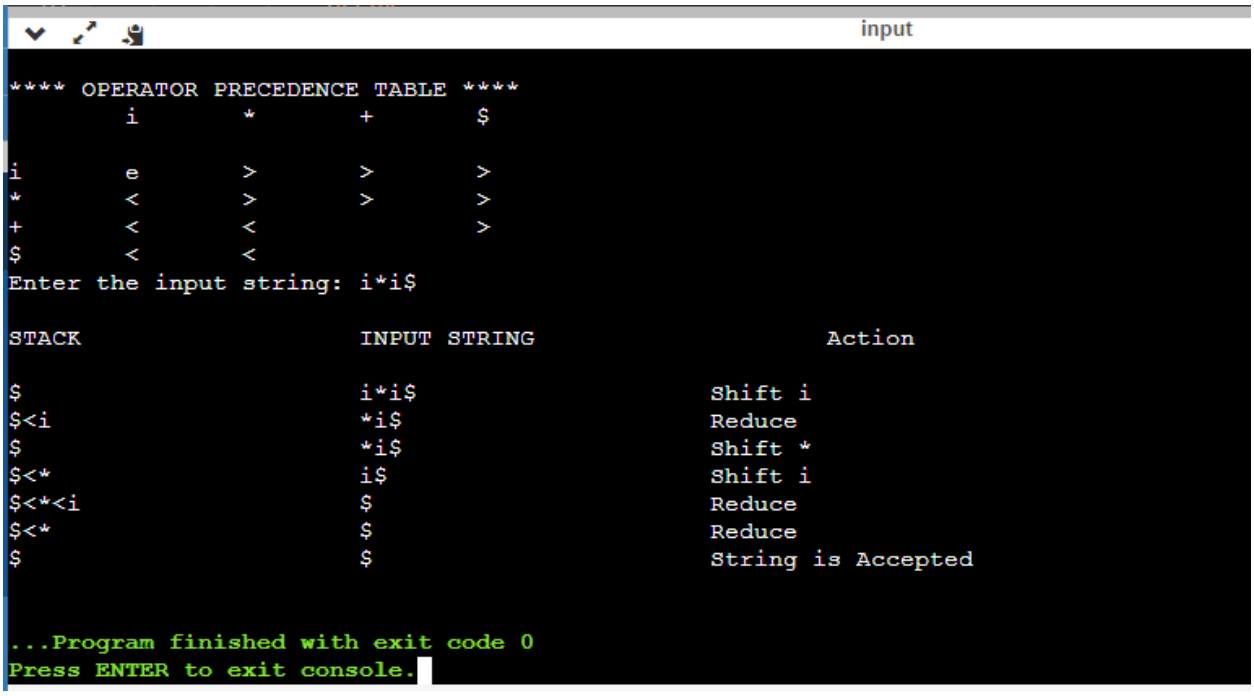
PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
   char stack[20],ip[20],opt[10][10][1],ter[]={"i*+$"};
   int i,j,k,n=4,top=0,col,row;
   for(i=0;i<n;i++)
   {
      stack[i]=NULL;  ip[i]=NULL;
      for(j=0;j<n;j++)
      {
         if(ter[i]=='i'&&ter[j]=='i')
         {
            opt[i][j][0]='e';
         }
         else if(ter[i]=='i')
            opt[i][j][0]='>';
         else if(ter[j]=='i')
            opt[i][j][0]='<';
         else if(ter[i]=='*')
            opt[i][j][0]='>';
         else if(ter[j]=='*')
            opt[i][j][0]='<';
         else if(ter[i]=='$')
            opt[i][j][0]='< ';
         else if(ter[i]=='$'&&ter[j]=='$')
            opt[i][j][0]='a';
         else if(ter[j]=='$')
            opt[i][j][0]='>';
      }
```

```
}
printf("\n**** OPERATOR PRECEDENCE TABLE ****\n");
for(i=0;i<n;i++)
{
   printf("\t%c",ter[i]);
}
printf("\n");
for(i=0;i<n;i++)
{
   printf("\n%c",ter[i]);
   for(j=0;j<n;j++)
   {
      printf("\t%c",opt[i][j][0]);
   }
}
stack[top]='$';
printf("\nEnter the input string: ");
scanf("%s",ip);
i=0;
printf("\nSTACK\t\t\tINPUT STRING\t\t\tAction\n");
printf("\n%s\t\t\t%s\t\t\t",stack,ip);
while(i<=strlen(ip)){
   for(k=0;k<n;k++)
   {
      if(stack[top]==ter[k])
         col=k;
      if(ip[i]==ter[k])
         row=k;
   }
   if((stack[top]=='$')&&(ip[i]=='$')){
      printf("String is Accepted\n");
```

```
    break;
}
else if((opt[col][row][0]=='<')||(opt[col][row][0]=='='))
{
    stack[++top]=opt[col][row][0];
    stack[++top]=ip[i];
    printf("Shift %c",ip[i]);
    i++;
}
else{
    if(opt[col][row][0]=='>'){
        while(stack[top]!='<'){
            --top;
        }
        top=top-1;
        printf("Reduce");
    }
    else{
        printf("\nString is not accepted");
        break;
    }
}
printf("\n");
for(k=0;k<=top;k++){
    printf("%c",stack[k]);
}
printf("\t\t\t");
for(k=i;k<strlen(ip);k++){
    printf("%c",ip[k]);
}
printf("\t\t\t");}}
```

OUTPUT:


input
**** OPERATOR PRECEDENCE TABLE ****
i * + $
i e > > >
* < > > >
+ < < >
$ < <
Enter the input string: i*i$
STACK INPUT STRING Action
$ i*i$ Shift i
$<i *i$ Reduce
$ *i$ Shift *
$<* i$ Shift i
$<*<i $ Reduce
$<* $ Reduce
$ $ String is Accepted
...Program finished with exit code 0
Press ENTER to exit console.

PROGRAM :

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
int size=0;
void Insert();
void Display();
void Delete();
int Search(char lab[]);
void Modify();
struct SymbTab
{
   char label[10],symbol[10];
   int addr;
   struct SymbTab *next;
};
struct SymbTab *first,*last;
void main()
{
int op,y;
char la[10];
do
{
   printf("\n\tSYMBOL TABLE IMPLEMENTATION\n");
   printf("\n\t1.INSERT\n\t2.DISPLAY\n\t3.DELETE\n\t4.SEARCH\n\t5.MODIFY\n\t6.END\n
");
   printf("\n\tEnter your option : ");
   scanf("%d",&op);
   switch(op)
```

```
    {
    case 1:
       Insert();
       break;
    case 2:
       Display();
       break;
    case 3:
       Delete();
       break;
    case 4:
       printf("\n\tEnter the label to be searched : ");
       scanf("%s",la);
       y=Search(la);
       printf("\n\tSearch Result:");
    if(y==1)
       printf("\n\tThe label is present in the symbol table\n");
    else
       printf("\n\tThe label is not present in the symbol table\n");
       break;
    case 5:
       Modify();
       break;
    case 6:
       exit(0);
    }
}while(op<6);
getch();
}
void Insert()
{
```

```
    int n;
    char l[10];
    printf("\n\tEnter the label : ");
    scanf("%s",l);
    n=Search(l);
    if(n==1)
       printf("\n\tThe label exists already in the symbol table\n\tDuplicate can't be inserted");
    else
    {
       struct SymbTab *p;
       p=malloc(sizeof(struct SymbTab));
       strcpy(p->label,l);
       printf("\n\tEnter the symbol : ");
       scanf("%s",p->symbol);
       printf("\n\tEnter the address : ");
       scanf("%d",&p->addr);
       p->next=NULL;
    if(size==0)
    {
       first=p;
       last=p;
    }
    else
    {
       last->next=p;
       last=p;
    }
    size++;
    }
    printf("\n\tLabel inserted\n");
}
```

```
void Display()
{
   int i;
   struct SymbTab *p;
   p=first;
   printf("\n\tLABEL\t\tSYMBOL\t\tADDRESS\n");
   for(i=0;i<size;i++)
   {
      printf("\t%s\t\t%s\t\t%d\n",p->label,p->symbol,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("\n\tWhat do you want to modify?\n");
printf("\n\t1.Only the label\n\t2.Only the address\n\t3.Both the label and address\n");
printf("\tEnter your choice : ");
scanf("%d",&choice);
switch(choice)
{
case 1:
   printf("\n\tEnter the old label : ");
   scanf("%s",l);
   s=Search(l);
if(s==0)
   printf("\n\tLabel not found\n");
else
{
   printf("\n\tEnter the new label : ");
   scanf("%s",nl);
   for(i=0;i<size;i++)
   {
      if(strcmp(p->label,l)==0)
      strcpy(p->label,nl);
      p=p->next;
   }
   printf("\n\tAfter Modification:\n");
   Display();
}
break;
case 2:
   printf("\n\tEnter the label where the address is to be modified : ");
   scanf("%s",l);
   s=Search(l);
   if(s==0)
```

```
            printf("\n\tLabel not found\n");
        else
        {
            printf("\n\tEnter the new address : ");
            scanf("%d",&add);
            for(i=0;i<size;i++)
            {
                if(strcmp(p->label,l)==0)
                    p->addr=add;
                    p=p->next;
            }
            printf("\n\tAfter Modification:\n");
            Display();
        }
        break;
    case 3:
        printf("\n\tEnter the old label : ");
        scanf("%s",l);
        s=Search(l);
        if(s==0)
            printf("\n\tLabel not found\n");
        else
        {
            printf("\n\tEnter the new label : ");
            scanf("%s",nl);
            printf("\n\tEnter the new address : ");
            scanf("%d",&add);
            for(i=0;i<size;i++)
            {
                if(strcmp(p->label,l)==0)
                {
```

```
                strcpy(p->label,nl);
                p->addr=add;
              }
              p=p->next;
            }
            printf("\n\tAfter Modification:\n");
            Display();
        }
        break;
}
}
void Delete()
{
    int a;
    char l[10];
    struct SymbTab *p,*q;
    p=first;
    printf("\n\tEnter the label to be deleted : ");
    scanf("%s",l);
    a=Search(l);
    if(a==0)
        printf("\n\tLabel 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)
            {
```

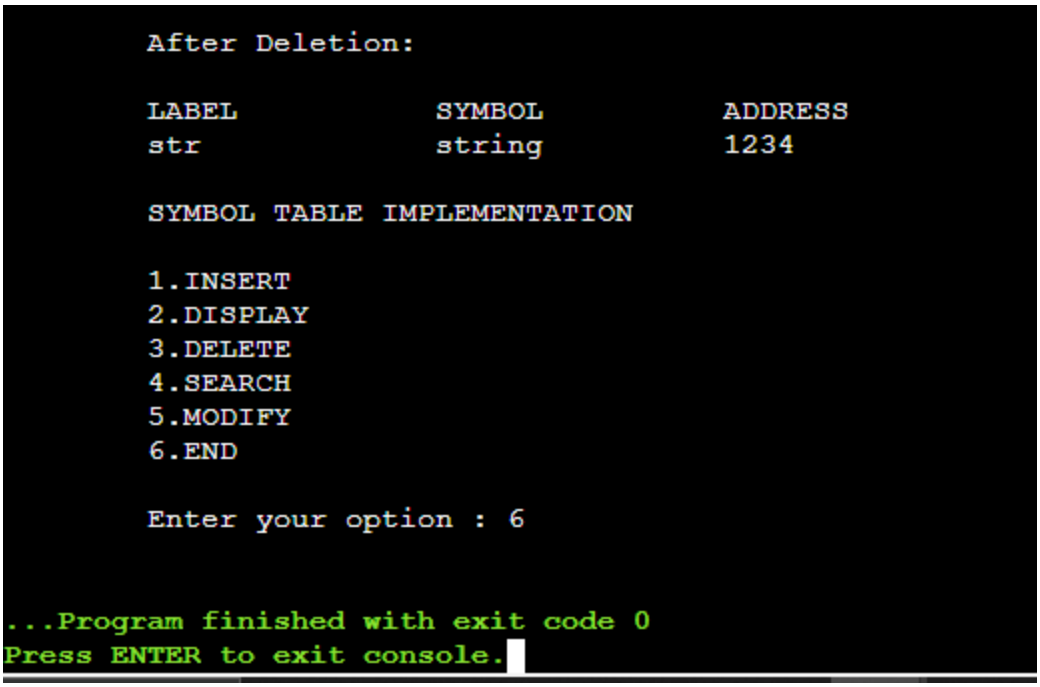
```
            p=p->next;
            q=q->next;
        }
        p->next=NULL;
        last=p;
    }
    else
    {
        q=p->next;
        while(strcmp(q->label,l)!=0)
        {
            p=p->next;
            q=q->next;
        }
        p->next=q->next;
    }
    size--;
    printf("\n\tAfter Deletion:\n");
    Display();
}
}
```

OUTPUT :

```
SYMBOL TABLE IMPLEMENTATION

1.INSERT
2.DISPLAY
3.DELETE
4.SEARCH
5.MODIFY
6.END

Enter your option : 1

Enter the label : str

Enter the symbol : string

Enter the address : 1234

Label inserted

SYMBOL TABLE IMPLEMENTATION

1.INSERT
2.DISPLAY
3.DELETE
4.SEARCH
5.MODIFY
6.END

Enter your option : 1

Enter the label : i

Enter the symbol : int

Enter the address : 4567
```

```
Label inserted

SYMBOL TABLE IMPLEMENTATION

1.INSERT
2.DISPLAY
3.DELETE
4.SEARCH
5.MODIFY
6.END

Enter your option : 2

LABEL           SYMBOL          ADDRESS
str             string          1234
i               int             4567

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 : i

Search Result:
The label is present 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
3.Both the label and address
Enter your choice : 1

Enter the old label : i

Enter the new label : i1

After Modification:

LABEL           SYMBOL          ADDRESS
str             string          1234
i1              int             4567

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 : i1
```


After Deletion:
LABEL SYMBOL ADDRESS
str string 1234
SYMBOL TABLE IMPLEMENTATION
1.INSERT
2.DISPLAY
3.DELETE
4.SEARCH
5.MODIFY
6.END
Enter your option : 6
...Program finished with exit code 0
Press ENTER to exit console.

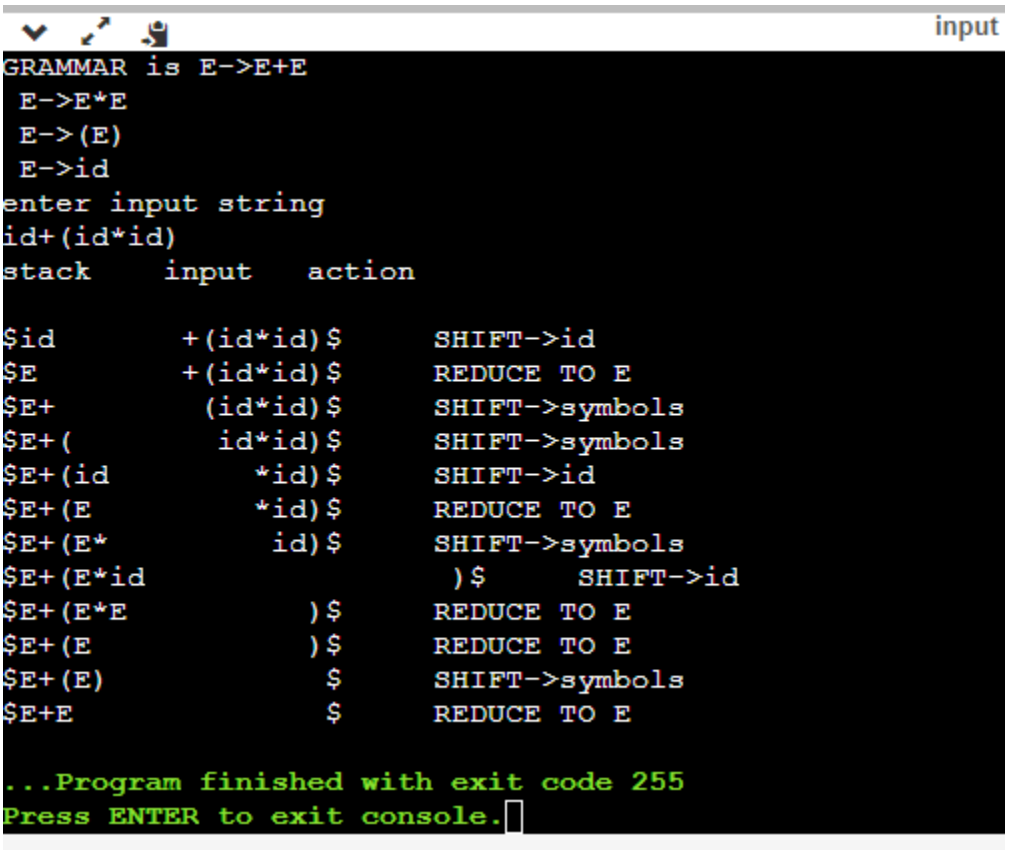
PROGRAM:

```
#include<stdio.h>
#include<string.h>
int k=0,z=0,i=0,j=0,c=0;
char a[16],ac[20],stk[15],act[10];
void check();
void main()
  {
    puts("GRAMMAR is E->E+E \n E->E*E \n E->(E) \n E->id");
    puts("enter input string ");
    gets(a);
    c=strlen(a);
    strcpy(act,"SHIFT->");
    puts("stack \t input \t action");
    for(k=0,i=0; j<c; k++,i++,j++)
     {
        if(a[j]=='i' && a[j+1]=='d')
        {
            stk[i]=a[j];
            stk[i+1]=a[j+1];
            stk[i+2]='\0';
            a[j]=' ';
            a[j+1]=' ';
                    printf("\n$%s\t%s$\t%sid",stk,a,act);
             check();
        }
        else
       {
           stk[i]=a[j];
           stk[i+1]='\0';
```

```
            a[j]=' ';
            printf("\n$%s\t%s$\t%ssymbols",stk,a,act);
            check();
          }
       }
       getch();
   }
void check()
   {
        strcpy(ac,"REDUCE TO E");
        for(z=0; z<c; z++)
        if(stk[z]=='i' && stk[z+1]=='d')
         {
            stk[z]='E';
            stk[z+1]='\0';
            printf("\n$%s\t%s$\t%s",stk,a,ac);
            j++;
}
    for(z=0; z<c; z++)
      if(stk[z]=='E' && stk[z+1]=='+' && stk[z+2]=='E')
        {
            stk[z]='E';
            stk[z+1]='\0';
            stk[z+2]='\0';
            printf("\n$%s\t%s$\t%s",stk,a,ac);
            i=i-2;
        }
    for(z=0; z<c; z++)
      if(stk[z]=='E' && stk[z+1]=='*' && stk[z+2]=='E')
        {
           stk[z]='E';
```

```
        stk[z+1]='\0';
        stk[z+1]='\0';
        printf("\n$%s\t%s$\t%s",stk,a,ac);
        i=i-2;
    }
for(z=0; z<c; z++)
  if(stk[z]=='(' && stk[z+1]=='E' && stk[z+2]==')')
    {
        stk[z]='E';
        stk[z+1]='\0';
        stk[z+1]='\0';
        printf("\n$%s\t%s$\t%s",stk,a,ac);
        i=i-2;
  }
```

OUTPUT:


input
GRAMMAR is E->E+E
E->E*E
E->(E)
E->id
enter input string
id+(id*id)
stack input action
$id +(id*id)$ SHIFT->id
$E +(id*id)$ REDUCE TO E
$E+ (id*id)$ SHIFT->symbols
$E+(id*id)$ SHIFT->symbols
$E+(id *id)$ SHIFT->id
$E+(E *id)$ REDUCE TO E
$E+(E* id)$ SHIFT->symbols
$E+(E*id)$ SHIFT->id
$E+(E*E)$ REDUCE TO E
$E+(E)$ REDUCE TO E
$E+(E) $ SHIFT->symbols
$E+E $ REDUCE TO E
...Program finished with exit code 255
Press ENTER to exit console.

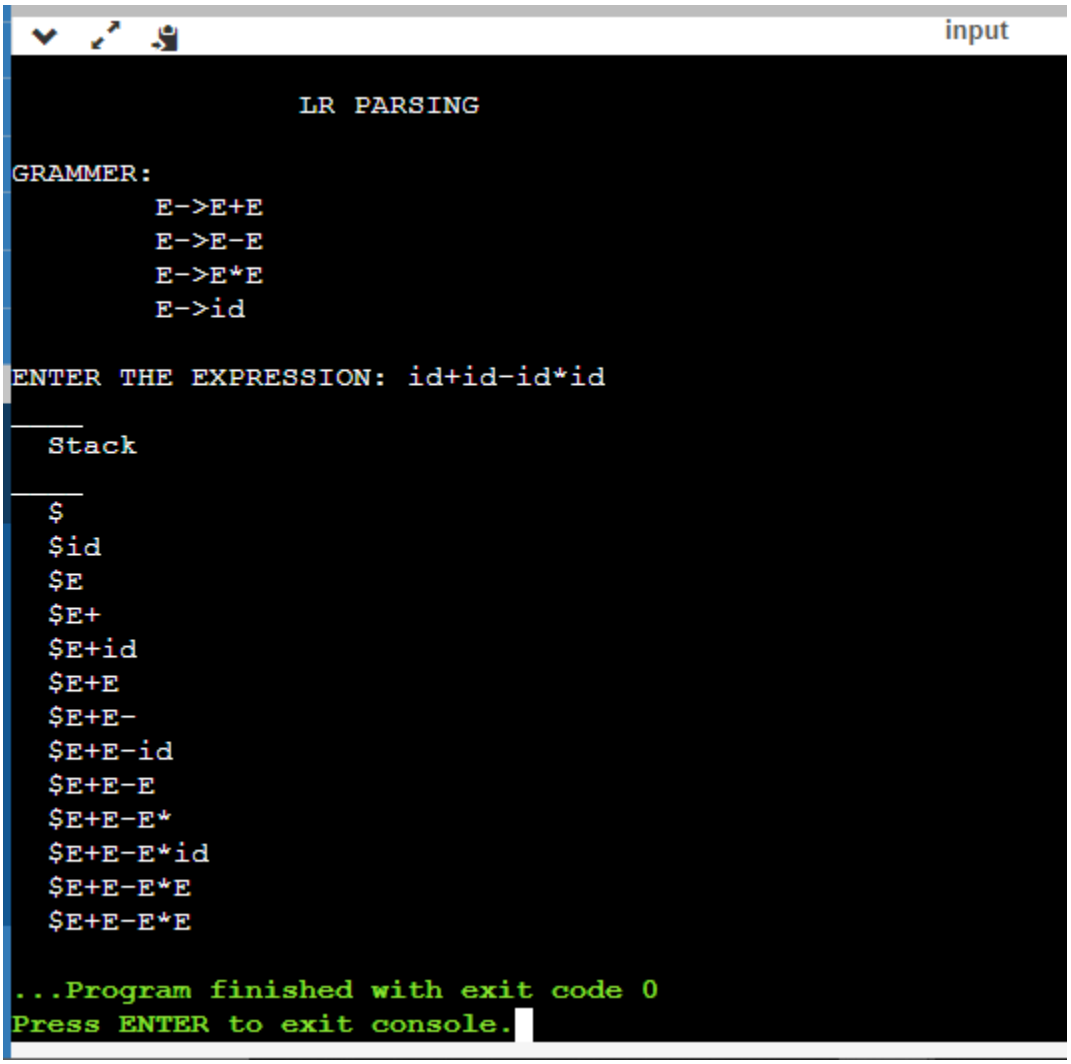
PROGRAM:

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

char stack[30];
int top=-1;

void push(char c)
{
   top++;
   stack[top]=c;
}

char pop()
{
   char c;
   if(top!=-1)
   {
      c=stack[top];
      top--;
      return c;
   }
   return'x';
}

void printstat()
{
   int i;
   printf("\n  $");
   for(i=0;i<=top;i++)
```

```
        printf("%c",stack[i]);
}

void main()
{
    int i,j,k,l;
    char s1[20],s2[20],ch1,ch2,ch3;
    printf("\n\t\tLR PARSING\n");
    printf("\nGRAMMER:");
    printf("\n\tE->E+E \n\tE->E-E");
    printf("\n\tE->E*E \n\tE->id");
    printf("\n\nENTER THE EXPRESSION: ");
    scanf("%s",s1);
    l=strlen(s1);
    j=0;
    printf("____");
    printf("\n  Stack\n");
    printf("____");
    printf("\n  $");
    for(i=0;i<=l;i++)
    {
        if(s1[i]=='i' && s1[i+1]=='d')
        {
            s1[i]=' ';
            s1[i+1]='E';
            printstat();
            printf("id");
            push('E');
            printstat();
        }
        else if(s1[i]=='+'||s1[i]=='-'||s1[i]=='*' ||s1[i]=='/' ||s1[i]=='d')
```

```
    {
        push(s1[i]);
        printstat();
    }
}
printstat();
l=strlen(s2);
while(l)
{
    ch1=pop();
    if(ch1=='x')
    {
        printf("\n  $");
        break;
    }
    if(ch1=='+'||ch1=='/'||ch1=='*'||ch1=='-')
    {
        ch3=pop();
        if(ch3!='E')
        {
            printf("error");
            exit(0);
        }
        else
        {
            push('E');
            printstat();
        }
    }
    ch2=ch1;
} getch();}
```

OUTPUT:


input
LR PARSING
GRAMMER:
E->E+E
E->E-E
E->E*E
E->id
ENTER THE EXPRESSION: id+id-id*id
Stack
$
$id
$E
$E+
$E+id
$E+E
$E+E-
$E+E-id
$E+E-E
$E+E-E*
$E+E-E*id
$E+E-E*E
$E+E-E*E
...Program finished with exit code 0
Press ENTER to exit console.

PROGRAM:

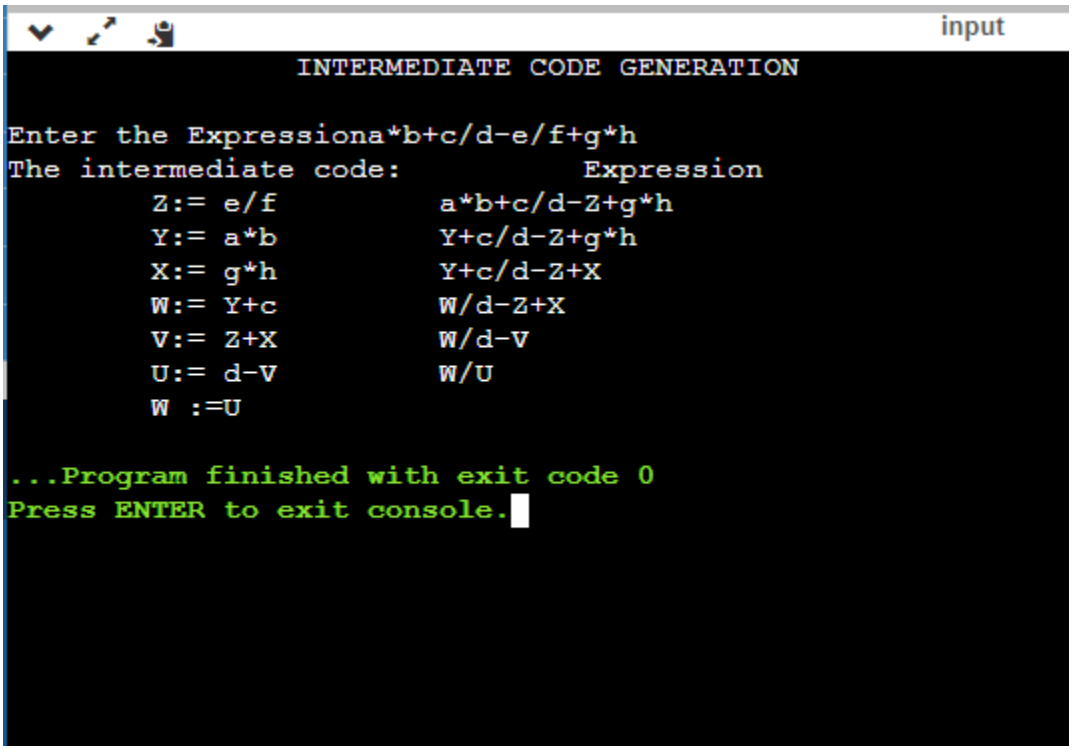
```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int i=1,j=0,no=0,tmpch=90;
char str[100],left[15],right[15];
void findopr();
void explore();
void fleft(int);
void fright(int);
struct exp{
   int pos;
   char op;
}k[15];
void main()
{
   printf("\t\tINTERMEDIATE CODE GENERATION\n\n");
   printf("Enter the Expression");
   scanf("%s",str);
   printf("The intermediate code:\t\tExpression\n");
   findopr();
   explore();
}
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++;
    }
}
```

OUTPUT:

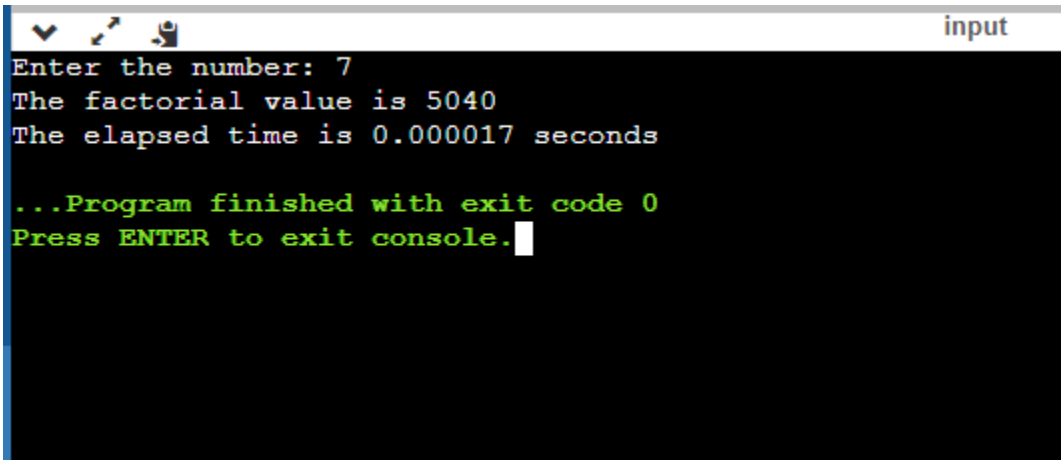

input
INTERMEDIATE CODE GENERATION
Enter the Expressiona*b+c/d-e/f+g*h
The intermediate code: Expression
Z:= e/f a*b+c/d-Z+g*h
Y:= a*b Y+c/d-Z+g*h
X:= g*h Y+c/d-Z+X
W:= Y+c W/d-Z+X
V:= Z+X W/d-V
U:= d-V W/U
W :=U
...Program finished with exit code 0
Press ENTER to exit console.

Before Optimization:

PROGRAM:

```
#include<stdio.h>
int main()
{
   int i,n;
   int fact=1;
   printf("Enter a number: ");
   scanf("%d",&n);
   for(i=n;i>=1;i--)
   {
      fact=fact*i;
   }
   printf("The factorial value is %d",fact);
   return 0;
}
```

OUTPUT:

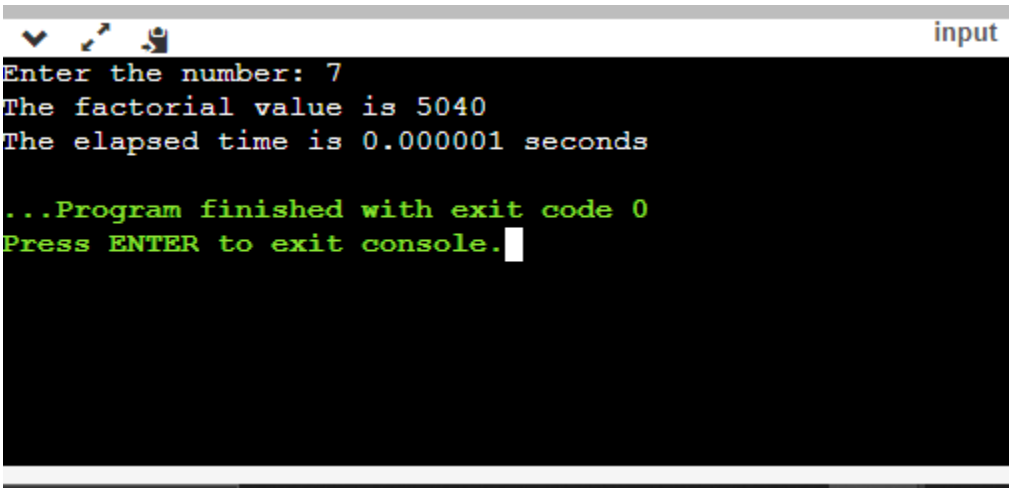

input
Enter the number: 7
The factorial value is 5040
The elapsed time is 0.000017 seconds
...Program finished with exit code 0
Press ENTER to exit console.

After optimization:

PROGRAM:

```
#include<stdio.h>
int main()
{
   int n,fact;
   fact=1;
   printf("Enter the number: ");
   scanf("%d",&n);
   do{
      fact=fact*n;
      n--;
   }while(n>0);
   printf("The factorial value is %d",fact);
   return 0;
}
```

OUTPUT:


input
Enter the number: 7
The factorial value is 5040
The elapsed time is 0.000001 seconds
...Program finished with exit code 0
Press ENTER to exit console.

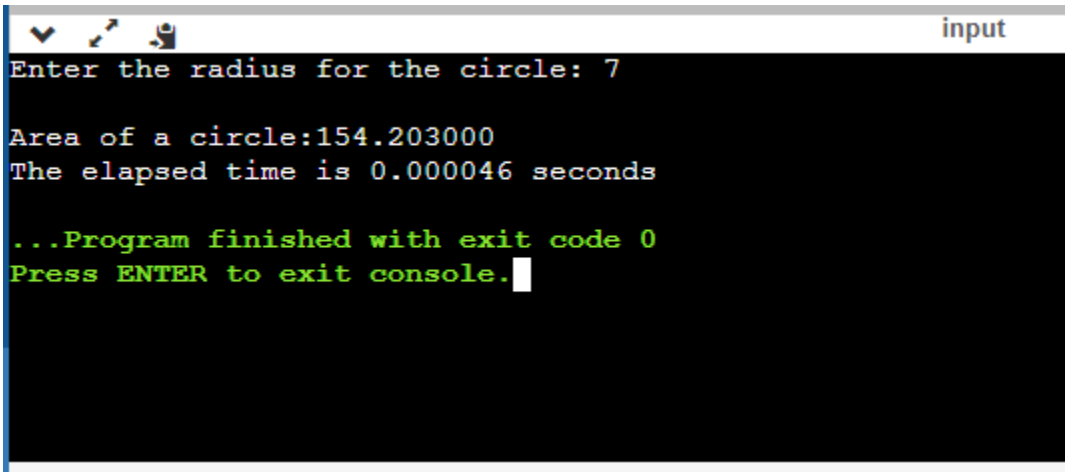
1.Area of a Circle

Before Optimization:

PROGRAM:

```
#include<stdio.h>
#define x 3.147
void main()
{
   float r,A;
   printf("Enter the radius for the circle: ");
   scanf("%f",&r);
   A=x*r*r;
   printf("\nArea of a circle:%f",A);
}
```

OUTPUT:

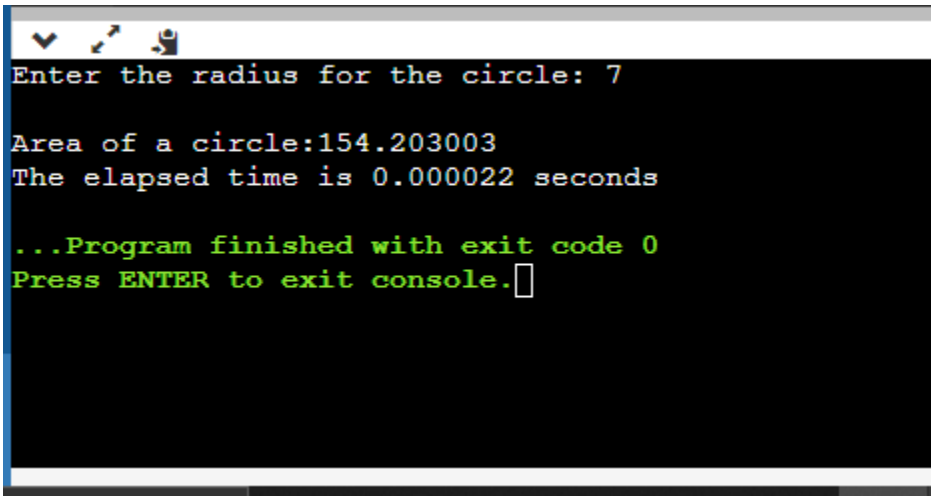

input
Enter the radius for the circle: 7
Area of a circle:154.203000
The elapsed time is 0.000046 seconds
...Program finished with exit code 0
Press ENTER to exit console.

After Optimization:

PROGRAM:

```
#include<stdio.h>
void main()
{
   float r;
   printf("Enter the radius for the circle: ");
   scanf("%f",&r);
   printf("\nArea of a circle:%f",3.147*r*r);
}
```

OUTPUT:


Enter the radius for the circle: 7
Area of a circle:154.203003
The elapsed time is 0.000022 seconds
...Program finished with exit code 0
Press ENTER to exit console.