# COMPILER DESIGN LAB

| S.NO | NAME OF THE EXPERIMENT |
|---|---|
| 1 | Write a LEX Program to scan reserved word & Identifiers of C Language. |
| 2 | Write a Program to compute the FIRST of a given grammar |
| 3 | Write a Program to compute the FOLLOW of a given grammar |
| 4 | Write a Program to construct PREDICTIVE LL(1) TABLE |
| 5 | Implement Predictive Parsing algorithm |
| 6 | Write a C program to generate three address code. |
|  |  |

**PROGRAM 1:**
**AIM: To Write a LEX Program to scan reserved word & Identifiers of C Language.**

```
/* program name is lexp.l */
%{
/* program to recognize a c program */
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;}
/*{printf("\n\n\t%s is a COMMENT\n",yytext);}*/
```

```
"*/" {COMMENT = 0;}
/* printf("\n\n\t%s is a COMMENT\n",yytext);}*/
{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;
}
```

**Input:**

```
$vi var.c
#include<stdio.h>
main()
{
int a,b;
}
$lex lex.l
$cc lex.yy.c
$./a.out var.c
```

**OUTPUT:**

```
#include<stdio.h> is a PREPROCESSOR DIRECTIVE
FUNCTION
main ()
BLOCK BEGINS
int is a KEYWORD
a IDENTIFIER
b IDENTIFIER
BLOCK ENDS
```

**PROGRAM 2:**

**Aim: To  Write a Program to compute the FIRST of a given grammar**

```
#include<stdio.h>
#include<ctype.h>

int main()
{
    int i,n,j,k;
    char str[10][10],f;
    printf("Enter the number of productions\n");
    scanf("%d",&n);
    printf("Enter grammar\n");
    for(i=0;i<n;i++)
        scanf("%s",&str[i]);
    for(i=0;i<n;i++)
    {
        f= str[i][0];
        int temp=i;
        if(isupper(str[i][3]))
        {
repeat:
            for(k=0;k<n;k++)
            {
                if(str[k][0]==str[i][3])
                {
                    if(isupper(str[k][3]))
                    {
                        i=k;
                        goto repeat;
                    }
                    else
                    {
                        printf("First(%c)=%c\n",f,str[k][3]);
                    }
                }
            }
        }
        else
        {
            printf("First(%c)=%c\n",f,str[i][3]);
        }
        i=temp;
    }
}
```

**Output:**

```
$ ./a.out
Enter the number of productions
3
Enter grammar
S->AB
A->a
B->b
First(S)=a
First(A)=a
First(B)=b
```

**PROGRAM 3**

**AIM: To Write a Program to compute the FOLLOW of a given grammar**

```
#include<stdio.h>

main()
{
        int np,i,j,k;
        char prods[10][10],follow[10][10],Imad[10][10];
        printf("enter no. of productions\n");
        scanf("%d",&np);
        printf("enter grammar\n");
        for(i=0;i<np;i++)
        {
         scanf("%s",&prods[i]);
        }

        for(i=0; i<np; i++)
        {
                if(i==0)
                {
                         printf("Follow(%c) = $\n",prods[0][0]);//Rule1
                }
                for(j=3;prods[i][j]!='\0';j++)
                {
                        int temp2=j;
                        //Rule-2: production A->xBb then everything in first(b) is in follow(B)
                        if(prods[i][j] >= 'A' && prods[i][j] <= 'Z')
                        {
                                 if((strlen(prods[i])-1)==j)
                                {
                                        printf("Follow(%c)=Follow(%c)\n",prods[i][j],prods[i][0]);
                                 }
                                int temp=i;
                                char f=prods[i][j];
                                if(!isupper(prods[i][j+1])&&(prods[i][j+1]!='\0'))
                                printf("Follow(%c)=%c\n",f,prods[i][j+1]);
                                if(isupper(prods[i][j+1]))
                                {
                                        repeat:
                                        for(k=0;k<np;k++)
                                        {
                                                if(prods[k][0]==prods[i][j+1])
                                                {
                                                        if(!isupper(prods[k][3]))
```

```
                            {
                              printf("Follow(%c)=%c\n",f,prods[k][3]);
                            }
                            else
                            {
                                i=k;
                                j=2;
                                goto repeat;
                            }
                        }
                    }
                }
                i=temp;
            }
            j=temp2;
        }
    }
}
```

__Output:__

```
$ ./a.out
enter no. of productions
3
enter grammar
S->AB
A->a
B->b
Follow(S) = $
Follow(A)=b
Follow(B)=Follow(S)
```

**PROGRAM 4:**

**AIM: To Write a Program to construct PREDICTIVE LL(1) TABLE**

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

char prod[10][20],start[2];
char nonterm[10],term[10];
char input[10],stack[50];
int table[10][10];
int te,nte;
int n;

void main()
{
        clrscr();
        init();
        parse();
        getch();
}

init()
{
        int i,j;
        printf("\nNOTE:\n");
        printf("The terminals should be entered in single lower case letters,special symbol and\n");
        printf("non-terminals should be entered in single upper case letters.\n");
        printf("extends to symbol is '->' and epsilon symbol is '@' \n");
        printf("\nEnter the no. of terminals:");
        scanf("%d",&te);
        for(i=0;i<te;i++)
        {
                fflush(stdin);
                printf("Enter the terminal %d:",i+1);
                scanf("%c",&term[i]);
        }
        term[i]='$';
        printf("\nEnter the no. of non terminals:");
        scanf("%d",&nte);
        for(i=0;i<nte;i++)
        {
                fflush(stdin);
                printf("Enter the non-terminal %d:",i+1);
                scanf("%c",&nonterm[i]);
        }
        printf("\nEnter the no. of productions:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("Enter the production %d:",i+1);
                scanf("%s",prod[i]);
        }
        fflush(stdin);
        printf("\nEnter the start symbol:");
        scanf("%c",&start[0]);
        printf("\nEnter the input string:");
        scanf("%s",input);
        input[strlen(input)]='$';
        printf("\n\nThe productions are:");
        printf("\nProductionNo.Production");
        for(i=0;i<n;i++)
                printf("\n %d             %s",i+1,prod[i]);
        printf("\n\nEnter the parsing table:");
        printf("\n Enter the production number in the required entry as mentioned above.");
        printf("\n Enter the undefined entry or error of table as '0'\n\n");
        for(i=0;i<nte;i++)
        {
                for(j=0;j<=te;j++)
                {
                        fflush(stdin);
                        printf("Entry of table[%c,%c]:",nonterm[i],term[j]);
                        scanf("%d",&table[i][j]);
                }
        }
 }

parse()
{
        int i,j,prodno;
        int top=-1,current=0;
        stack[++top]='$';
        stack[++top]=start[0];
        do
        {
                if((stack[top]==input[current])&&(input[current]=='$'))
                {
                        printf("\nThe given input string is parsed");
                        getch();
                        exit(0);
                }
                else if(stack[top]==input[current])
                {
                        top--;
                        current++;
                }
                else if(stack[top]>='A'&&stack[top]<='Z')
                {
                        for(i=0;i<nte;i++)
                                if(nonterm[i]==stack[top]) break;
                        for(j=0;j<=te;j++)
                                if(term[j]==input[current]) break;
                        prodno=table[i][j];
                        if(prodno==0)
                        {
                                printf("\nThe given input string is not parsed");
                                getch();
                                exit(0);
                        }
                        else
                        {
                                for(i=strlen(prod[prodno-1])-1;i>=3;i--)
                                {
                                        if(prod[prodno-1][i]!='@')
                                        stack[top++]=prod[prodno-1][i];
                                }
                                top--;
                        }
                }
                else
                {
                        printf("\nThe given input string is not parsed");
                        getch();
                        exit(0);
                }
        }while(1);
}
```

**Input:**

```
NOTE:
The terminals should be entered in single lower case letters,special symbol and
non-terminals should be entered in single upper case letters.
extends to symbol is '->' and epsilon symbol is '@'

Enter the no. of terminals:2
Enter the terminal 1:a
Enter the terminal 2:b

Enter the no. of non terminals:3
Enter the non-terminal 1:S
Enter the non-terminal 2:A
Enter the non-terminal 3:B

Enter the no. of productions:7
Enter the production 1:S->aAB
Enter the production 2:S->bA
Enter the production 3:S->@
Enter the production 4:A->aAB
Enter the production 5:A->@
Enter the production 6:B->bB
Enter the production 7:B->@

Enter the start symbol:S

Enter the input string:aab$

The productions are:
ProductionNo.  Production
 1             S->aAB
 2             S->bA
 3             S->@
 4             A->aAB
 5             A->@
 6             B->bB
 7             B->@

Enter the parsing table:
 Enter the production number in the required entry as mentioned above.
 Enter the undefined entry or error of table as '0'


Entry of table[S,a]:1
Entry of table[S,b]:2
Entry of table[S,$]:3
Entry of table[A,a]:4
Entry of table[A,b]:5
Entry of table[A,$]:5
Entry of table[B,a]:0
Entry of table[B,b]:6
Entry of table[B,$]:7
```

**<u>Output:</u>**

```
The given input string is parsed
```

**PROGRAM 5:**

## AIM: To Implement Predictive Parsing algorithm.

```
#include<stdio.h>
#include<conio.h>
char nt[]={'E','A','T','B','F'},ter[]={'i','+','*','(',')','$'};
char arr[20][20][20]={
{"TA"," "," ","TA"," "," "},
{" ","+TA"," "," ","#","#"},
{"FB"," "," ","FB"," "," "},
{" ","#","*FB"," ","#","#"},
{"i"," "," ","(E)"," "," "}
};
char ipstr[20];
char stack[40],prod[10];
int i=0,top=1,ia,ix;
void main(void )
{
void pop();
void push(char );
int resolve_nt(char );
int resolve_t(char );
void advance();
char a,x;
int len,temp,k;
stack[0]='$';
stack[1]='E';
printf("Enter the input string:\n");
printf("Enter $ as an end marker\n");
scanf("%s",ipstr);
printf("I/P String\t\tStack Contents\t\tProduction Used\n");

while(1)
{
a=ipstr[i];
```

```
x=stack[top];
/*To display the input string*/
for(k=i;ipstr[k]!='$';k++)
printf("%c",ipstr[k]);
printf("$\t\t");
if(x==a)
{
if(x=='$')
{
printf("\rinput string is accepted");
break;

}
else
{
pop();
advance();
}
}
else if(isupper(x))
{
ix=resolve_nt(x);
ia=resolve_t(a);
strcpy(prod,arr[ix][ia]);
len=strlen(prod);
pop();
for(k=1;k<=len;k++)
push(prod[len-k]);
if(stack[top]=='#')
pop();
}
else
{
printf("Error: Could not parse teh input string");
break;
}
/*To display the stack contents and the production used*/
for(k=0;k<=top;k++)
printf("%c",stack[k]);
printf("\t\t\t\t%s\n",prod);
}
getch();
}
void push(char t)
{
top+=1;
stack[top]=t;
}
void pop()
{
top--;
}
void advance()
{
i++;
}
int resolve_nt(char t)
{
int k,index;
for(k=0;k<5;k++)
{
if(t==nt[k])
{
index=k;
break;
}
}
return index;
}
```

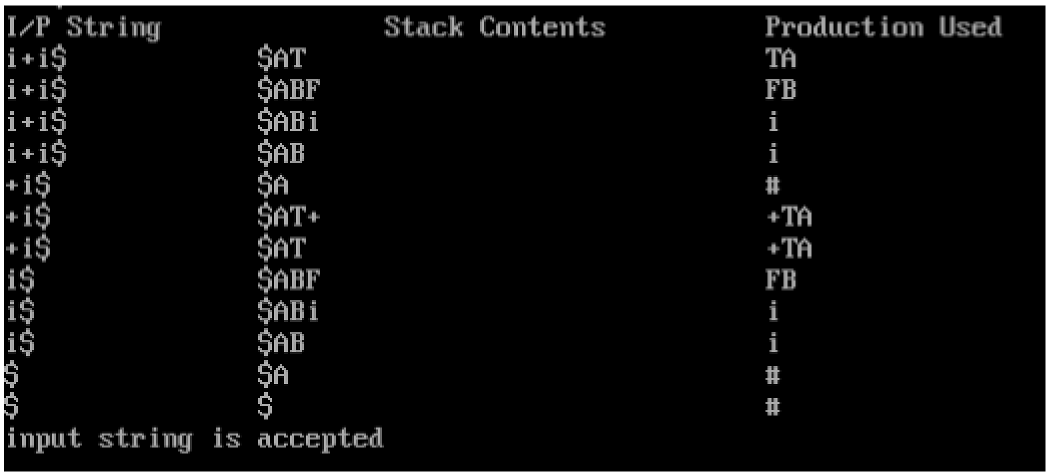
```
int resolve_t(char t)
{
int k,index;
for(k=0;k<6;k++)
{
if(t==ter[k])
{
index=k;
break;
}
}
return index;
}
```

**INPUT:**

**Enter a string**

**i+i$**

**OUTPUT:**



```
I/P String                Stack Contents              Production Used
i+i$            $AT                                   TA
i+i$            $ABF                                  FB
i+i$            $ABi                                  i
i+i$            $AB                                   i
+i$             $A                                    #
+i$             $AT+                                  +TA
+i$             $AT                                   +TA
i$              $ABF                                  FB
i$              $ABi                                  i
i$              $AB                                   i
$               $A                                    #
$               $                                     #
input string is accepted
```

## PROGRAM 6:

## AIM: To write a C program to generate three address code.

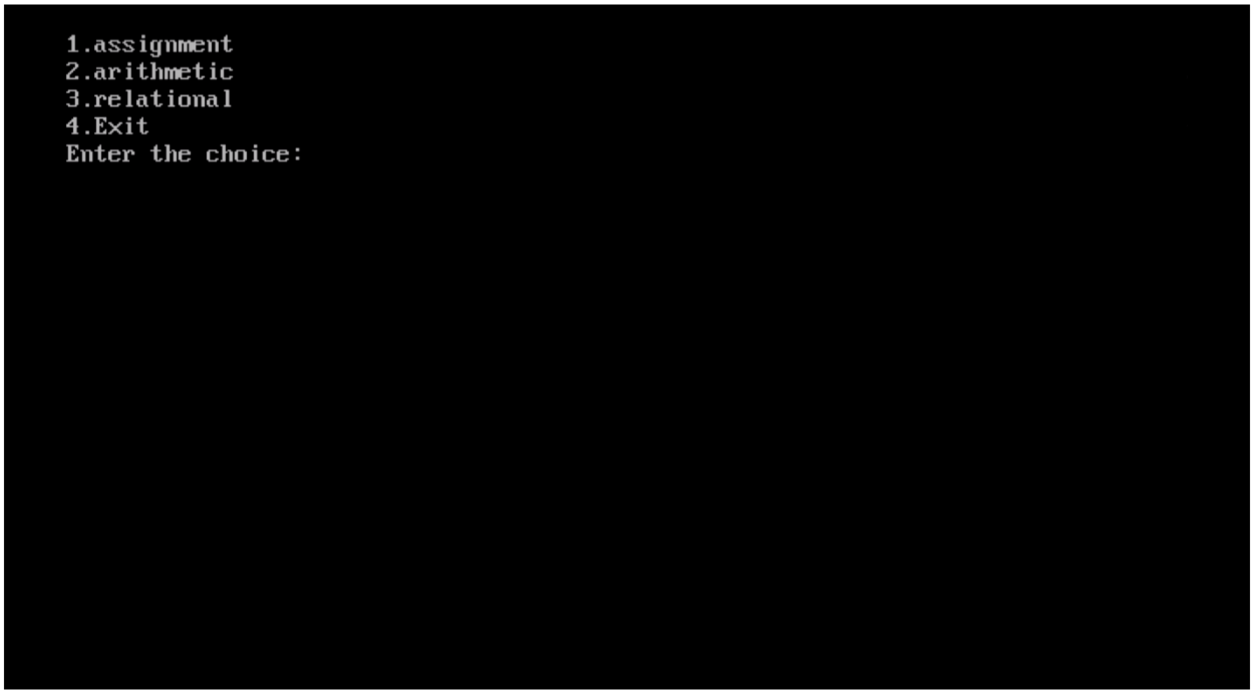
```
#include<stdio.h>
 #include<string.h>
#include<iostream>
 void pm();
void plus();
void div();
 int i,ch,j,l,addr=100;
 char ex[10], exp[10] ,exp1[10],exp2[10],id1[5],op[5],id2[5];
 int main()
{
while(1)
 {
 printf("\n1.assignment\n2.arithmetic\n3.relational\n4.Exit\nEnter the choice:");
scanf("%d",&ch);
switch(ch)
 {
 case 1: printf("\nEnter the expression with assignment operator:");
scanf("%s",exp);
l=strlen(exp);
 exp2[0]='\0';
 i=0;
while(exp[i]!='=')
{
i++;
}
strncat(exp2,exp,i);
strrev(exp);
exp1[0]='\0';
strncat(exp1,exp,l-(i+1));
```

```
strrev(exp1);
printf("Three address code:\ntemp=%s\n%s=temp\n",exp1,exp2);
break;
case 2:
printf("\nEnter the expression with arithmetic operator:");
scanf("%s",ex);
strcpy(exp,ex);
 l=strlen(exp);
exp1[0]='\0';
 for(i=0;i<l;i++)
{
if(exp[i]=='+'||exp[i]=='-')
 {
if(exp[i+2]=='/'||exp[i+2]=='*')
 {
pm();
break;
 }
 else
{
plus();
break;
}
 }
else if(exp[i]=='/'||exp[i]=='*')
{
div();
break;
}
 }
 break;
case 3: printf("Enter the expression with relational operator");
 scanf("%s%s%s",&id1,&op,&id2);
if(((strcmp(op,"<")==0)||(strcmp(op,">")==0)||(strcmp(op,"<=")==0)||(strcmp(op,">=")==0)||(strcmp(
op,"==")==0)||(strcmp(op,"!=")==0))==0)
```

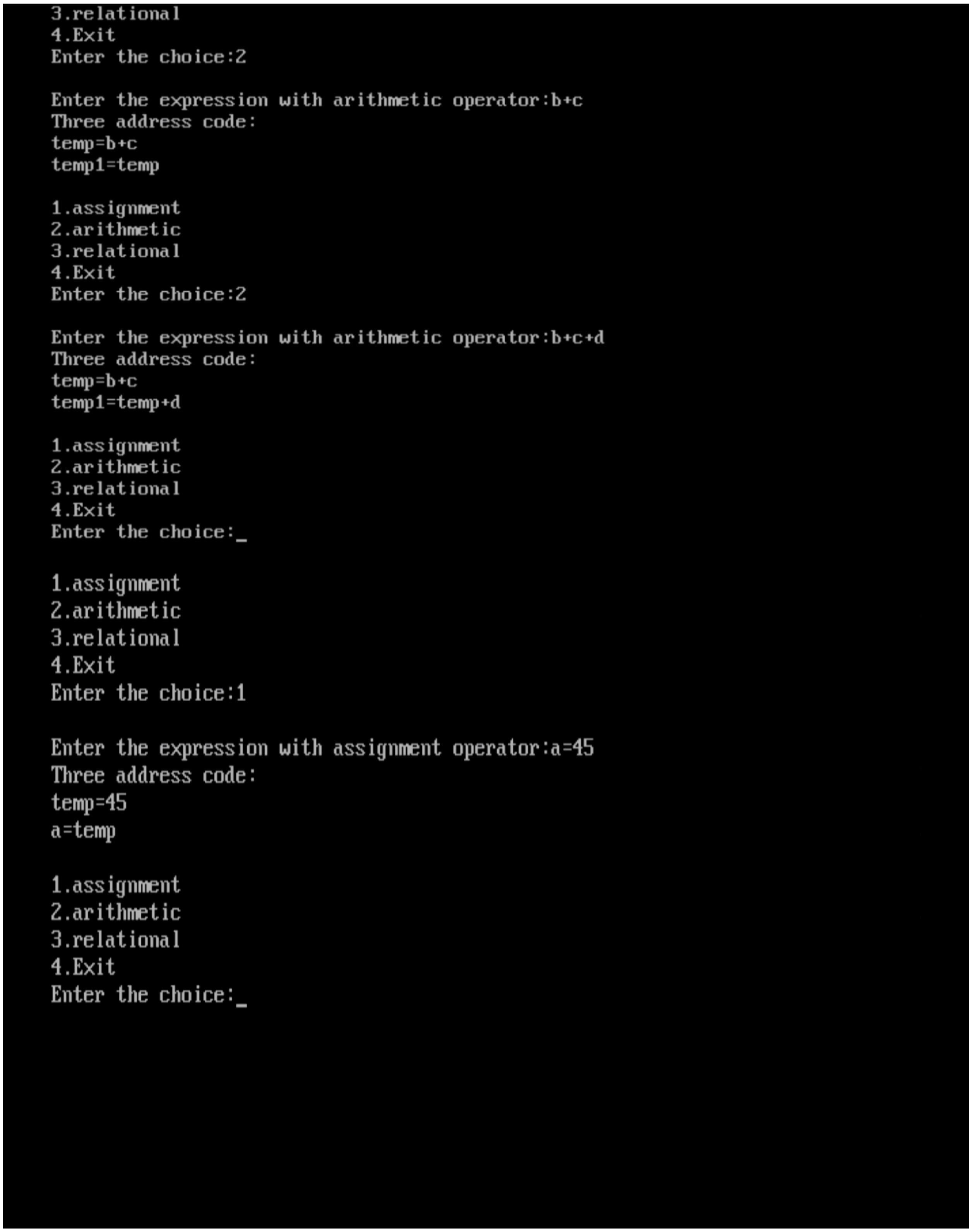
```
printf("Expression is error");
else
{



printf("\n%d\tif %s%s%s goto %d",addr,id1,op,id2,addr+3);
addr++;
 printf("\n%d\t T:=0",addr);
addr++;
printf("\n%d\t goto %d",addr,addr+2);
addr++;
 printf("\n%d\t T:=1",addr);
 }
 break;
 case 4:
exit(0);
 }


}
 }
void pm()
{
 strrev(exp);
 j=l-i-1;
strncat(exp1,exp,j);
strrev(exp1);
printf("Three address code:\ntemp=%s\ntemp1=%c%ctemp\n",exp1,exp[j+1],exp[j]);
 }
 void div()
 {
 strncat(exp1,exp,i+2);
 printf("Three address code:\ntemp=%s\ntemp1=temp%c%c\n",exp1,exp[i+2],exp[i+3]);
 }
 void plus()
 {
```

```
strncat(exp1,exp,i+2);
printf("Three address code:\ntemp=%s\ntemp1=temp%c%c\n",exp1,exp[i+2],exp[i+3]);
}
```

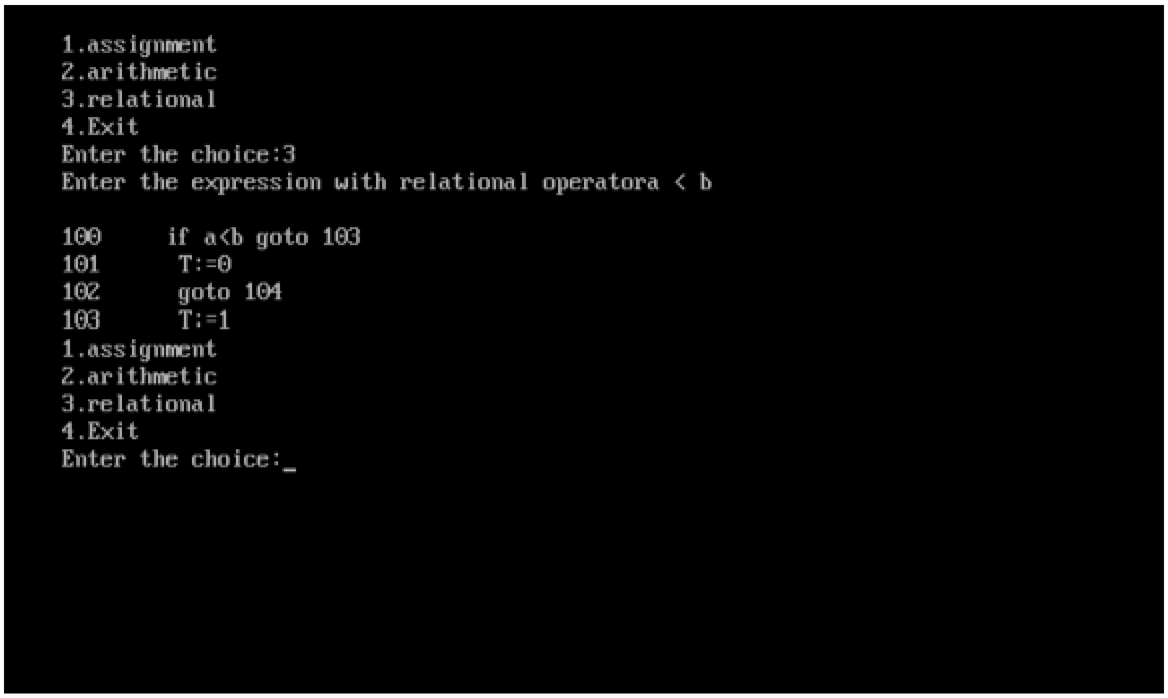
**Output**



```
1.assignment
2.arithmetic
3.relational
4.Exit
Enter the choice:
```



```
3.relational
4.Exit
Enter the choice:2

Enter the expression with arithmetic operator:b+c
Three address code:
temp=b+c
temp1=temp

1.assignment
2.arithmetic
3.relational
4.Exit
Enter the choice:2

Enter the expression with arithmetic operator:b+c+d
Three address code:
temp=b+c
temp1=temp+d

1.assignment
2.arithmetic
3.relational
4.Exit
Enter the choice:_

1.assignment
2.arithmetic
3.relational
4.Exit
Enter the choice:1

Enter the expression with assignment operator:a=45
Three address code:
temp=45
a=temp

1.assignment
2.arithmetic
3.relational
4.Exit
Enter the choice:_
```



```
1.assignment
2.arithmetic
3.relational
4.Exit
Enter the choice:3
Enter the expression with relational operatora < b

100     if a<b goto 103
101      T:=0
102      goto 104
103      T:=1
1.assignment
2.arithmetic
3.relational
4.Exit
Enter the choice:_
```