EX. NO:1	**DEVELOP A LEXICAL ANALYZER TO RECOGNIZE A FEW PATTERNS IN C AND CREATE SYMBOL TABLE WHILE RECOGNIZING IDENTIFIERS**
Date:	

PROGRAM(Pattern recognition):

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
#include<string.h>
void main()
{
FILE *fi,*fo,*fop,*fk;
int flag=0,i=1;
char c,t,a[15],ch[15],file[20];
clrscr();
printf("\n Enter the File Name:");
scanf("%s",&file);
fi=fopen(file,"r");
fo=fopen("input.c","");
fop=fopen("oper.c","r");
fk=fopen("key.c","r");
c=getc(fi);
while(!feof(fi))
{
if(isalpha(c)||isdigit(c)||(c=='['||c==']'||c=='.'==1))
fputc(c,fo);
else
{
if(c=='\n')
fprintf(fo,"\t$\t");
else fprintf(fo,"\t%c\t",c);
}
c=getc(fi);
```

```
}
fclose(fi);
fclose(fo);
fi=fopen("input.c","r");
printf("\n Lexical Analysis");
fscanf(fi,"%s",a);
printf("\n Line: %d\n",i++);
while(!feof(fi))
{
if(strcmp(a,"$")==0)
{
printf("\n Line: %d \n",i++);
fscanf(fi,"%s",a);
}
fscanf(fop,"%s",ch);
while(!feof(fop))
{
if(strcmp(ch,a)==0)
{
fscanf(fop,"%s",ch);
printf("\t\t%s\t:\t%s\n",a,ch);
flag=1;
} fscanf(fop,"%s",ch);
}
rewind(fop);
fscanf(fk,"%s",ch);
while(!feof(fk))
{
if(strcmp(ch,a)==0)
```

```
{
fscanf(fk,"%k",ch);
printf("\t\t%s\t:\tKeyword\n",a);
flag=1;
}
fscanf(fk,"%s",ch);
}
rewind(fk);
if(flag==0)
{
if(isdigit(a[0]))
printf("\t\t%s\t:\tConstant\n",a);
else
printf("\t\t%s\t:\tIdentifier\n",a);
}
flag=0;
fscanf(fi,"%s",a); }
getch();
}
```

Key.C:

```
int
void
main
char
if
for
while
else
printf
```

scanf

FILE

Include stdio.h

conio.h

iostream.h

Oper.C:

(open para

) closepara

{ openbrace } closebrace < lesser

> greater

" doublequote ' singlequote : colon

; semicolon

preprocessor = equal

== asign

% percentage ^ bitwise

& reference * star

+ add - sub

\ backslash / slash

Input.C:

```
#include  "stdio.h"
#include "conio.h"
void main()
{
int a=10,b,c;
a=b*c;
getch();
}
```

PROGRAM(Symbol Table):

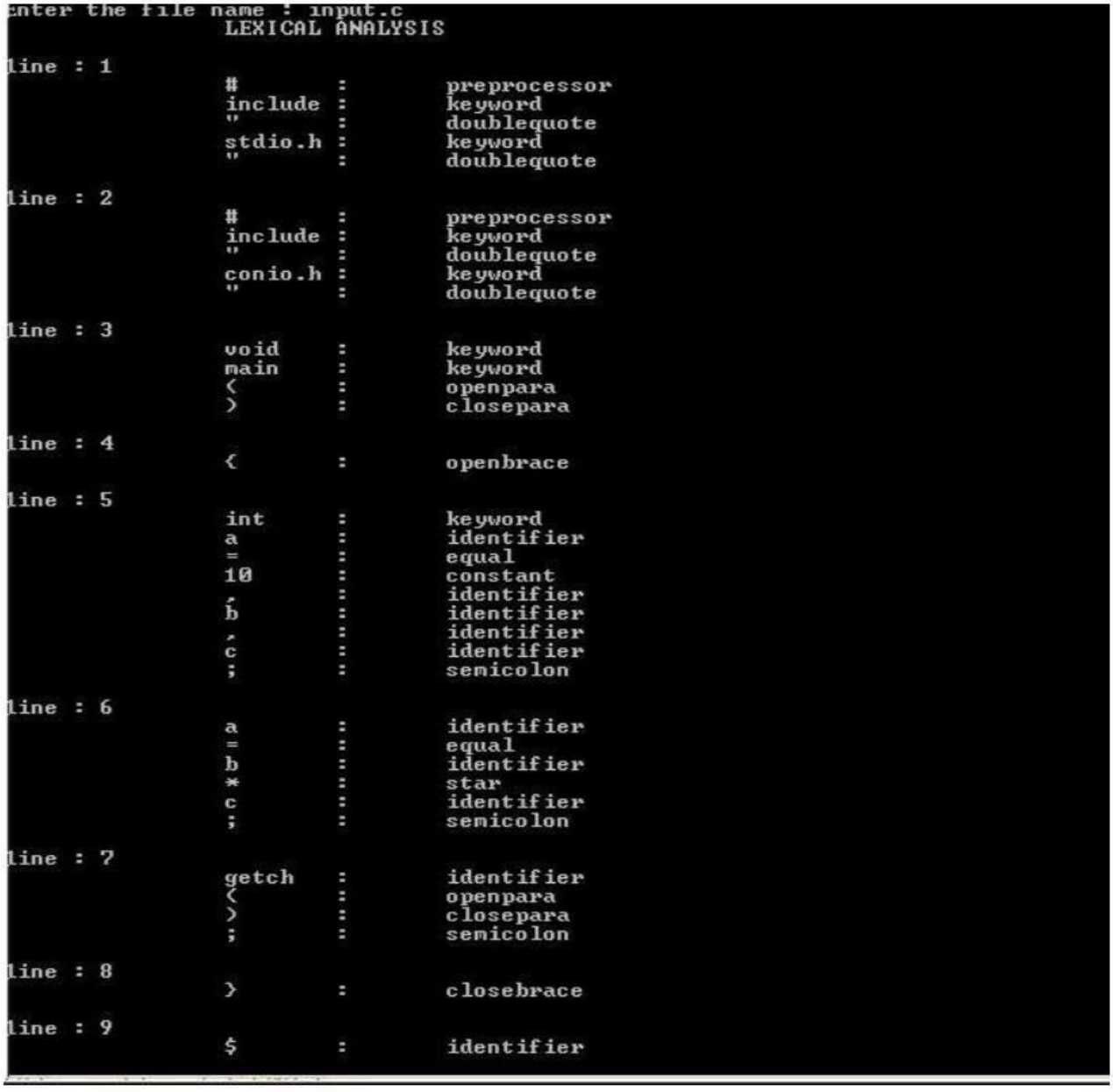
```
#include<stdio.h>
```

```
#include<conio.h>a
#include<malloc.h>
#include<string.h>
#include<math.h>
#include<ctype.h>
void main()
{
int i=0,j=0,x=0,n,flag=0; void *p,*add[15]; char ch,srch,b[15],d[15],c; //clrscr();
printf("expression terminated by $:"); while((c=getchar())!='$') {
b[i]=c; i++;
}
n=i-1;
printf("given expression:");
i=0;
while(i<=n)
{
printf("%c",b[i]); i++;
}
printf("symbol table\n");
printf("symbol\taddr\ttype\n");
while(j<=n)
{
c=b[j]; if(isalpha(toascii(c)))
{
if(j==n)
{
p=malloc(c); add[x]=p;
d[x]=c;
printf("%c\t%d\tidentifier\n",c,p);
```

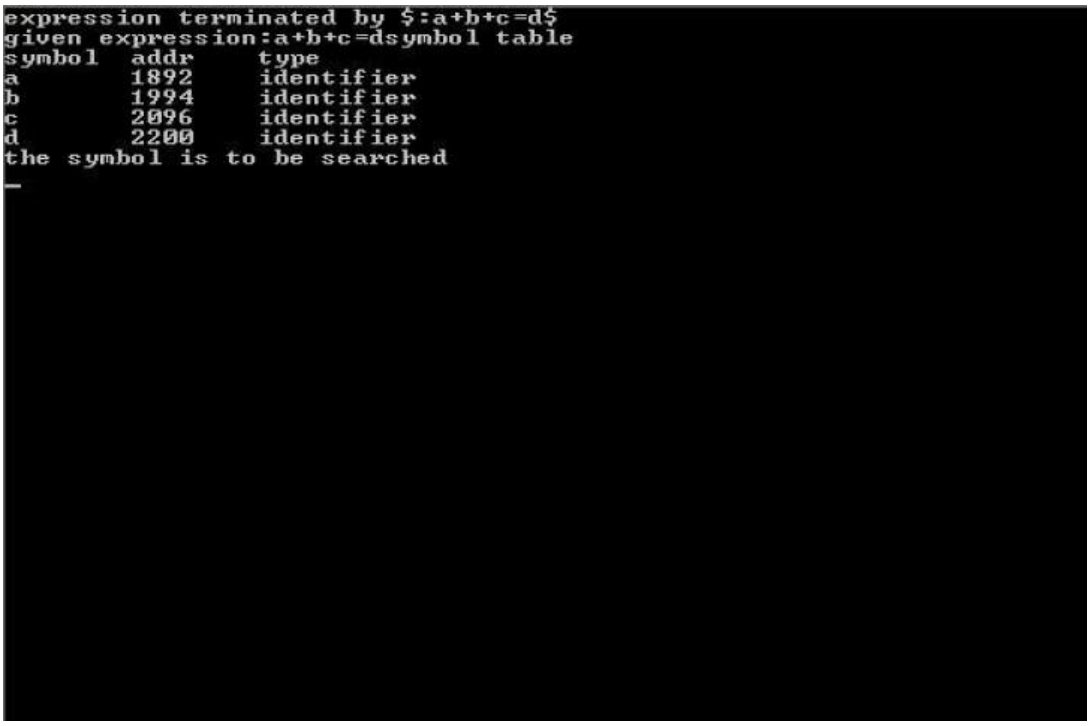
```
}
else
{
ch=b[j+1];
if(ch=='+'||ch=='-'||ch=='*'||ch=='=')
{
p=malloc(c);
add[x]=p;
d[x]=c;
printf("%c\t%d\tidentifier\n",c,p);
x++;
}
}
} j++;
}
printf("the symbol is to.besearched\n");
srch=getch();
for(i=0;i<=x;i++)
{
if(srch==d[i])
{
printf("symbol found\n");
printf("%c%s%d\n",srch,"@address",add[i]);
flag=1;
}
}
if(flag==0)
printf("symbol not found\n");
getch();
```

}

OUTPUT(Pattern recognition):



OUTPUT(Symbol Table):



EX. NO:2	**IMPLEMENTATION OF LEXICAL ANALYZER USING LEX TOOL**
Date:	

PROGRAM:

```
#include<stdio.h>
#include<ctype.h>
#include<conio.h>
#include<string.h>
char vars[100][100];
int vcnt;
char input[1000],c;
char token[50],tlen;
int state=0,pos=0,i=0,id;
char *getAddress(char str[])
{
for(i=0;i<vcnt;i++)
if(strcmp(str,vars[i])==0)
return vars[i];
strcpy(vars[vcnt],str);
return vars[vcnt++];
}
int isrelop(char c)
{
if(c=='+'||c=='-'||c=='*'||c=='/'||c=='%'||c=='^')
return 1;
else
return 0;
}
int main(void)
{
clrscr();
printf("Enter the Input String:");
```

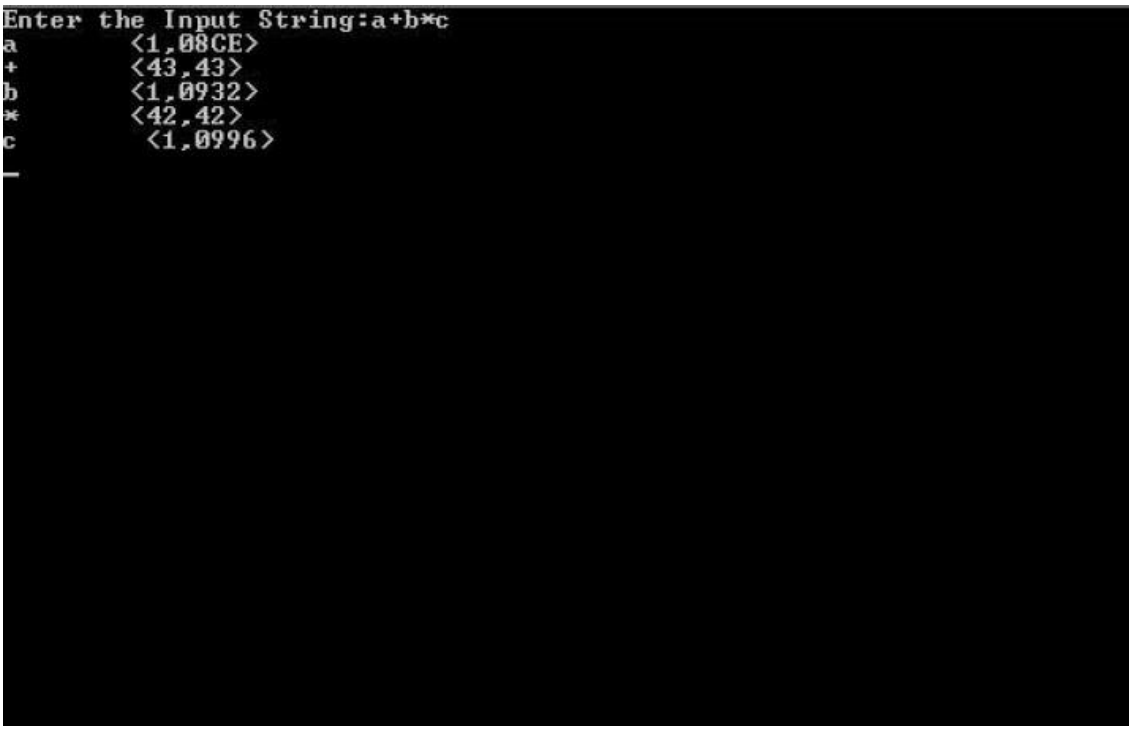
```
gets(input);
do
{
c=input[pos];
putchar(c);
switch(state)
{
case 0:
if(isspace(c))
printf("\b");
if(isalpha(c))
{
token[0]=c;
tlen=1;
state=1;
}
if(isdigit(c))
state=2;
if(isrelop(c))
state=3;
if(c==';')
printf("\t<3,3>\n");
if(c=='=')
printf("\t<4,4>\n");
break;
case 1:
if(!isalnum(c))
{
token[tlen]='\o';
```

```
printf("\b\t<1,%p>\n",getAddress(token));
state=0;
pos--;
}
else
token[tlen++]=c;
break;
case 2:
if(!isdigit(c))
{
printf("\b\t<2,%p>\n",&input[pos]);
state=0;
pos--;
}
break;
case 3:
id=input[pos-1];
if(c=='=')
printf("\t<%d,%d>\n",id*10,id*10);
else{
printf("\b\t<%d,%d>\n",id,id);
pos--;
}state=0;
break;
}
pos++;
}
while(c!=0);
getch();
```

return 0;

}

OUTPUT



EX. NO:3(A)	**GENERATE YACC SPECIFICATION FOR A FEW SYNTACTIC CATEGORIES**
Date:	**A) ARITHMETIC EXPRESSION RECOGNIZER**

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{ char s[5]; clrscr();
printf("\n Enter any operator:"); gets(s);
switch(s[0])
{
case'>': if(s[1]=='=')
printf("\n Greater than or equal"); else
printf("\n Greater than"); break;
case'<': if(s[1]=='=')
printf("\n Less than or equal");
else
printf("\nLess than");
break;
case'=':
if(s[1]=='=')
printf("\nEqual to");
else
printf("\nAssignment");
break;
case'!':
if(s[1]=='=')
printf("\nNot Equal");
else
printf("\n Bit Not");
break;
case'&':
```

```
if(s[1]=='&')
printf("\nLogical AND");
else
printf("\n Bitwise AND");
break;
case'|':
if(s[1]=='|')
printf("\nLogical OR");
else
printf("\nBitwise OR");
break;
case'+':
printf("\n Addition");
break;
case'-':
printf("\nSubstraction");
break;
case'*':
printf("\nMultiplication");
break;
case'/':
printf("\nDivision");
break;
case'%': printf("Modulus");
break;
default:
printf("\n Not a operator");
}
getch(); }
```

OUTPUT:

EX. NO:3(B)	**B) PROGRAM TO RECOGNISE A VALID VARIABLE WHICH STARTS WITH A LETTER FOLLOWED BY ANY NUMBER OF LETTERS OR DIGITS**
Date:	

PROGRAM :

variable_test.l

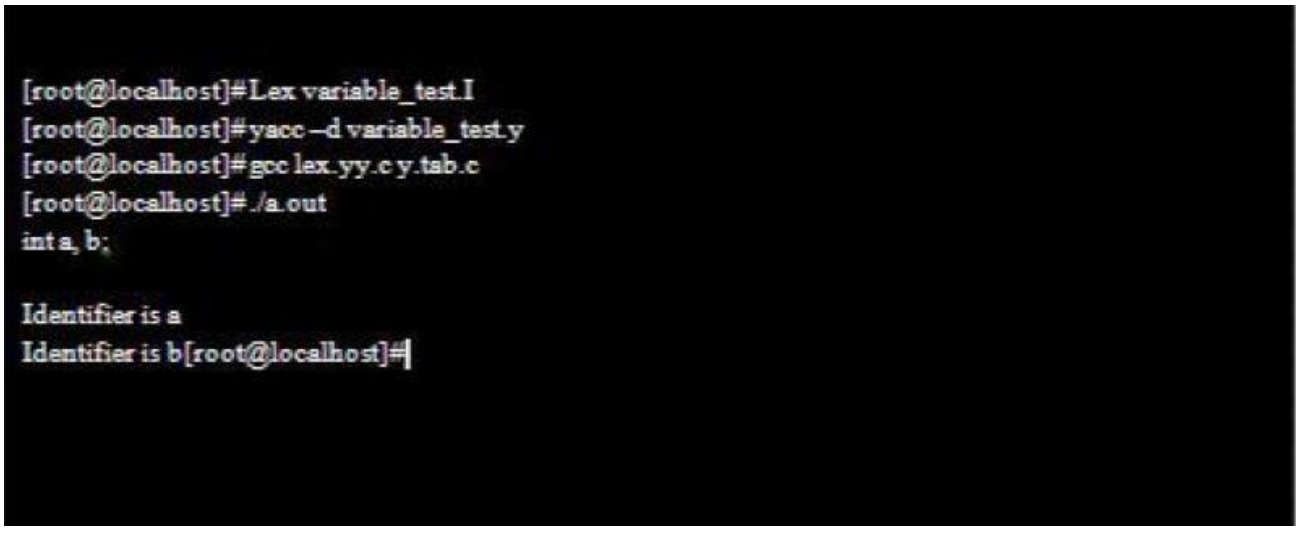
```
%{
/* This LEX program returns the tokens for the Expression */
#include "y.tab.h"
%}
%%
"int " {return INT;}
"float" {return FLOAT;}
"double" {return DOUBLE;}
[a-zA-Z]*[0-9]*{
printf("\nIdentifier is %s",yytext);
return ID;
}
return yytext[0];
\n return 0;
int yywrap()
{
return 1;
}
```

variable_test.y

```
%{
#include
/* This YACC program is for recognising the Expression*/ %}
%token ID INT FLOAT DOUBLE
%%
D;T L
;
L:L,ID
```

```
|ID
;
T:INT
|FLOAT
|DOUBLE
;
%%
extern FILE *yyin;
main()
{
do
{
yyparse();
}while(!feof(yyin));
}
yyerror(char*s)
{
}
```

OUTPUT:



EX. NO:3(C)	**C) PROGRAM TO RECOGNISE A VALID CONTROL STRUCTURES SYNTAX OF C LANGUAGE (FOR LOOP, WHILE LOOP, IF-ELSE, IF-ELSE-IF, SWITCH-CASE, ETC.)**
Date:	

PROGRAM:

```
#include <stdio.h>
#include <conio.h>  // For Turbo C/C++
int main()
{
FILE *file;
char line[1000];
int insideComment = 0;
int insideString = 0;
int insideCharacter = 0;
clrscr(); // For Turbo C/C++
file = fopen("sample.c", "r");
if (file == NULL) {
printf("Unable to open the file.\n");
return 1;
}
printf("Valid Control Structures:\n");
while (fgets(line, sizeof(line), file) != NULL) {
for (int i = 0; line[i] != '\0'; i++) {
if (line[i] == '/' && line[i + 1] == '*') {
insideComment = 1;
i++;
}
else if (line[i] == '*' && line[i + 1] == '/') {
insideComment = 0;
} else if (line[i] == '/' && line[i + 1] == '/') {
break;  // Skip line comments
} else if (line[i] == '"') {
insideString = 1 - insideString;
```

```
} else if (line[i] == '\'') {
insideCharacter = 1 - insideCharacter;
} else if (!insideComment && !insideString && !insideCharacter) {
if (strncmp(line + i, "for", 3) == 0) {
printf("For loop: %s\n", line);
break;
} else if (strncmp(line + i, "while", 5) == 0) {
printf("While loop: %s\n", line);
break;
} else if (strncmp(line + i, "if", 2) == 0) {
printf("If statement: %s\n", line);
break;
} else if (strncmp(line + i, "else if", 7) == 0) {
printf("Else-if statement: %s\n", line);
break;
} else if (strncmp(line + i, "else", 4) == 0) {
printf("Else statement: %s\n", line);
break;
} else if (strncmp(line + i, "switch", 6) == 0) {
printf("Switch statement: %s\n", line);
break;
} else if (strncmp(line + i, "case", 4) == 0) {
printf("Case label: %s\n", line);
break;
} else if (strncmp(line + i, "default", 7) == 0) {
printf("Default label: %s\n", line);
break;
}
}
```

```
}
}
fclose(file);
getch();  // For Turbo C/C++
return 0;
}
```

Sample.c:

```
#include <stdio.h>
int main() {
int i;
for (i = 0; i < 10; i++) {
if (i % 2 == 0) {
printf("Even\n");
}
else{
printf("Odd\n");
}
}
return 0;
}
```

OUTPUT:

```
Valid Control Structures:
For loop:     for (i = 0; i < 10; i++) {
If statement:         if (i % 2 == 0) {
Else statement:         } else {
```

EX. NO:3(D)	**D) PROGRAM TO IMPLEMENT A CALCULATOR USING LEX AND YACC**
Date:	

PROGRAM:

```
%{
#include<stdio.h>
int op=0,i;
float a,b;
%}
dig[0-9]+|([0-9]*)"."([0-9]+)
add "+"
sub  "-"
mul"*"
div "/"
pow "^"
ln \n
%%
{dig}{digi();}
{add}{op=1;}
{sub}{op=2;}
{mul}{op=3;}
{div}{op=4;}
{pow}{op=5;}
{ln}{printf("\n the result:%f\n\n",a);}
%%
digi()
{
if(op==0)
a=atof(yytext);
else
{
b=atof(yytext);
```

```
switch(op)
{
case 1:a=a+b;
break;
case 2:a=a-b;
break;
case 3:a=a*b;
break;
case 4:a=a/b;
break;
case 5:for(i=a;b>1;b--)
a=a*i;
break;
}
op=0;
}
}
main(int argv,char *argc[])
{
yylex();
}
yywrap()
{
return 1;
}
```

OUTPUT:

Lex cal.l

Cc lex.yy.c-ll

a.out

4*8

The result=32

EX. NO:4	**GENERATE THREE ADDRESS CODE FOR A SIMPLE PROGRAM USING LEX AND YACC**
Date:	

PROGRAM:

LEX file:calc.l

```
%{%{
#include "y.tab.h"
%}

%%
int     { return INT; }
return  { return RETURN; }
[ \t]   { /* Ignore whitespace */ }
\n      { return EOL; }
.       { return yytext[0]; }

%%

int yywrap() {
   return 1;
}
```

YACC File:calc.y

```
%{
#include <stdio.h>
#include <stdlib.h>
%}

%token INT
%token RETURN
%token EOL

%%
program:
   INT main EOL statement_list return_statement
   ;
```

```
main:
   INT MAIN '(' ')' '{' '}'
   ;

statement_list:
   /* Empty */
   |
   statement statement_list
   ;

statement:
   INT ID '=' expr EOL  { printf("%s = %s %s %s\n", $2, $4, $1, $6); }
   ;

expr:
   ID     { $$ = $1; }
   |
   NUM     { $$ = $1; }
   |
   expr '+' expr  { $$ = $1 + $3; }
   |
   expr '-' expr  { $$ = $1 - $3; }
   ;

return_statement:
   RETURN NUM EOL   { printf("return %s\n", $2); }
   ;

%%

int main() {
```

```
    yyparse();
    return 0;
}
```

C program:

```
int main() {
    int a, b, c;
    a = 5;
    b = 3;
    c = a + b;
    return 0;
}
```

OUTPUT:

```
a = 5
b = 3
t1 = a + b
return 0
```

EX. NO:5	**IMPLEMENTATION OF TYPE CHECKING**
Date:	

PROGRAM:

```
#include<stdio.h>
char str[50],opstr[75];
int f[2][9]={2,3,4,4,4,0,6,6,0,1,1,3,3,5,5,0,5,0};
int col,col1,col2;
char c;
swt()
{
switch(c)
{
case'+':col=0;break;
case'-':col=1;break;
case'*':col=2;break;
case'/':col=3;break;
case'^':col=4;break;
case'(':col=5;break;
case')':col=6;break;
case'd':col=7;break;
case'$':col=8;break;
default:printf("\nTERMINAL MISSMATCH\n");
exit(1);
}
// return 0;
}
main()
{
int i=0,j=0,col1,cn,k=0;
int t1=0,foundg=0;
char temp[20];
```

```
clrscr();
printf("\nEnter arithmetic expression:");
scanf("%s",&str);
while(str[i]!='\0')
i++;
str[i]='$';
str[++i]='\0';
printf("%s\n",str);
come:
i=0;
opstr[0]='$';
j=1;
c='$';
swt();
col1=col;
c=str[i];
swt();
col2=col;
if(f[1][col1]>f[2][col2])
{
opstr[j]='>';
j++;
}
else if(f[1][col1]<f[2][col2])
{
opstr[j]='<';
j++;
}
else
```

```
{
opstr[j]='=';j++;
}
while(str[i]!='$')
{
c=str[i];
swt();
col1=col;
c=str[++i];
swt();
col2=col;
opstr[j]=str[--i];
j++;
if(f[0][col1]>f[1][col2])
{
opstr[j]='>';
j++;
}
else if(f[0][col1]<f[1][col2])
{
opstr[j]='<';
j++;
}
else
{
opstr[j]='=';j++;
}
i++;
}
```

```
opstr[j]='$';
opstr[++j]='\0';
printf("\nPrecedence Input:%s\n",opstr);
i=0;
j=0;
while(opstr[i]!='\0')
{
foundg=0;
while(foundg!=1)
{
if(opstr[i]=='\0')goto redone;
if(opstr[i]=='>')foundg=1;
t1=i;
i++;
}
if(foundg==1)
for(i=t1;i>0;i--)
if(opstr[i]=='<')break;
if(i==0){printf("\nERROR\n");exit(1);}
cn=i;
j=0;
i=t1+1;
while(opstr[i]!='\0')
{
temp[j]=opstr[i];
j++;i++;
}
temp[j]='\0';
opstr[cn]='E';
```

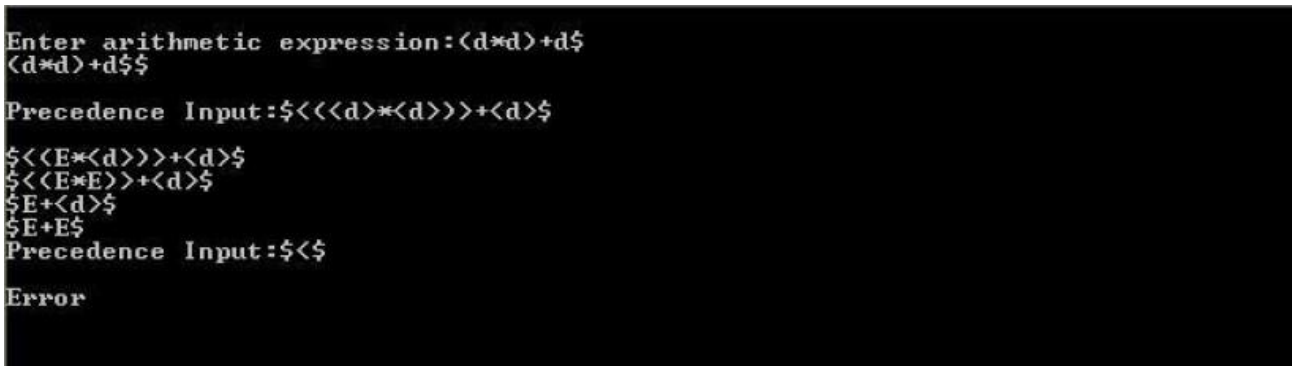
```
opstr[++cn]='\0';
strcat(opstr,temp);
printf("\n%s",opstr);
i=1;
}
redone:k=0;
while(opstr[k]!='\0')
{
k++;
if(opstr[k]=='<')
{
Printf("\nError");
exit(1);
}
}
if((opstr[0]=='$')&&(opstr[2]=='$'))goto sue;
i=1
while(opstr[i]!='\0')
{
c=opstr[i];
if(c=='+'||c=='*'||c=='/'||c=='$')
{
temp[j]=c;j++;}
i++;
}
temp[j]='\0';
strcpy(str,temp);
goto come;
sue:
```

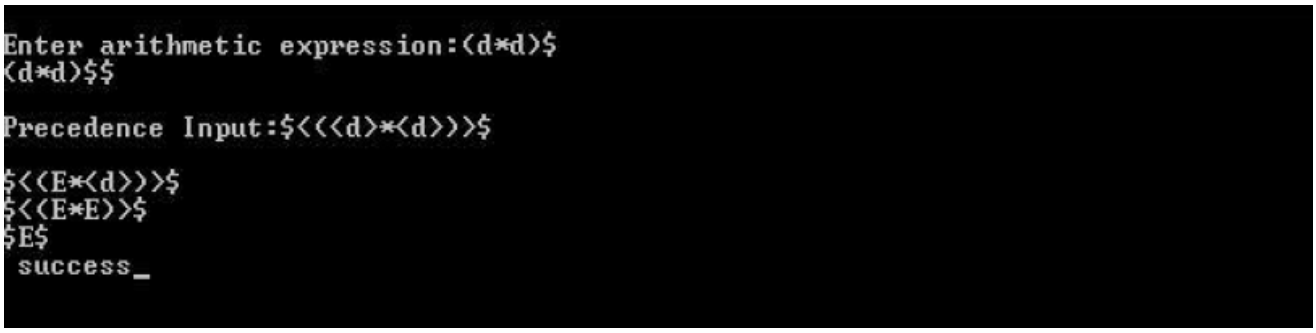
```
printf("\n success");

return 0;

}
```

OUTPUT:





EX. NO:6	**IMPLEMENTATION OF SIMPLE CODE OPTIMIZATION TECHNIQUES**
Date:	

PROGRAM:

Before:

Using for :

```
#include<iostream.h>
#include <conio.h>
int main()
{
int i, n;
int fact=1;
cout<<"\nEnter a number: ";
cin>>n;
for(i=n;i>=1;i--)
fact=fact *i;
cout<<"The factoral value is: "<<fact;
getch();
return 0;
}
```

After:

Using do-while:

```
#include<iostream.h>
#include<conio.h>
void main()
{
clrscr();
int n,f;
f=1;
cout<<"Enter the number:\n";
cin>>n;
do
```

```
{
f=f*n;
n--;
}while(n>0);
cout<<"The factorial value is:"<<f;
getch();
}
```

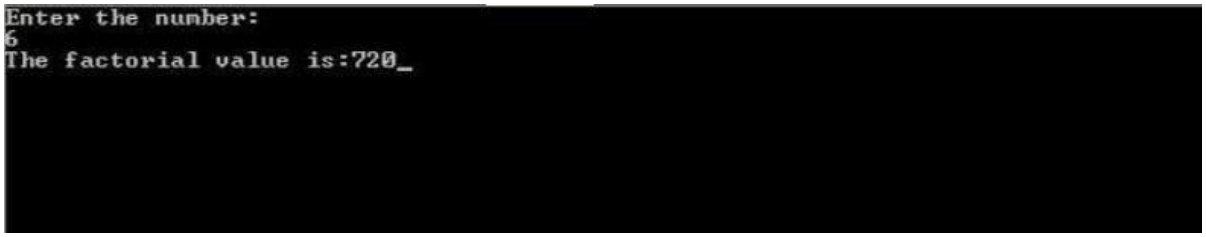
OUTPUT:

Before:

Using for :

After:

Using do-while:



EX. NO:7	**IMPLEMENT THE BACK END OF THE COMPILER**
Date:	

PROGRAM:

```
#include<stdio.h>
#include<stdio.h>
//#include<conio.h>
#include<string.h>
void main()
{
char icode[10][30],str[20],opr[10];
int i=0;
//clrscr();
printf("\n Enter the set of intermediate code (terminated by exit):\n");
do
{
scanf("%s",icode[i]);
} while(strcmp(icode[i++],"exit")!=0);
printf("\n target code generation");
printf("\n************************");
i=0;
do
{
strcpy(str,icode[i]);
switch(str[3])
{
case '+':
strcpy(opr,"ADD");
break;
case '-':
strcpy(opr,"SUB");
break;
```

```
case '*':
strcpy(opr,"MUL");
break;
case '/':
strcpy(opr,"DIV");
break;
}
printf("\n\tMov %c,R%d",str[2],i);
printf("\n\t%s%c,R%d",opr,str[4],i);
printf("\n\tMov R%d,%c",i,str[0]);
}while(strcmp(icode[++i],"exit")!=0);
//getch();
}
```

OUTPUT: