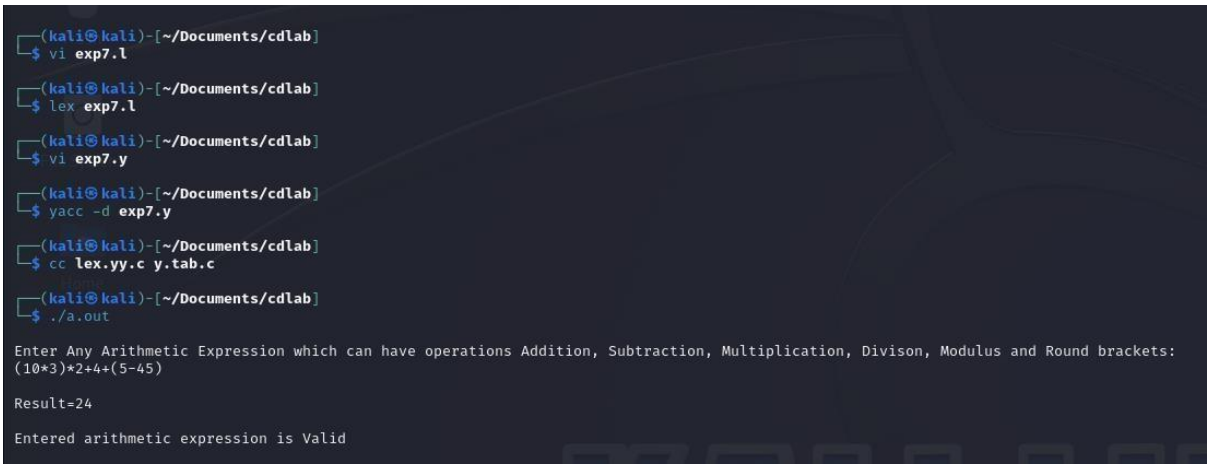
```
 |E'/'E {$$=$1/$3;}
 |E'%'E {$$=$1%$3;}
 |'('E')' {$$=$2;}
 | NUMBER {$$=$1;}
;
%%
void main(){
  printf("\nEnter Any Arithmetic Expression which can have operations Addition,
Subtraction, Multiplication, Divison, Modulus and Round brackets:\n");
  yyparse();   if(flag==0)    printf("\nEntered arithmetic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

OUTPUT:



RESULT:

Thus, arithmetic operations that takes digits,*, + using lex and yacc have been performed.

Roll Number: 210701075
Name: Harish R

Ex No: 8
Date:

GENERATE THREE ADDRESS CODES AIM:

AIM:

To generate three address code using C program.

ALGORITHM:

- Get address code sequence.
- Determine current location of 3 using address (for 1st operand).
- If the current location does not already exist, generate move (B, O).
- Update address of A (for 2nd operand).
- If the current value of B and () is null, exist.
- If they generate operator () A, 3 ADPR.
- Store the move instruction in memory.

PROGRAM:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
void pm(); void
plus(); void divi(); int
i,ch,j,l,addr=100;
char ex[10], exp0[10] ,exp1[10],exp22[10],id1[5],op[5],id2[5];
char *strrev(char *str){       char *p1, *p2;
     if (! str || ! *str)
return str;
     for (p1 = str, p2 = str + strlen(str) - 1; p2 > p1; ++p1, --p2){             *p1
^= *p2;
          *p2 ^= *p1;
          *p1 ^= *p2;
     }
return str;  }
void
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",exp0); l=strlen(exp0); exp22[0]='\0';
i=0;
```

```
while(exp0[i]!='=')
        i++;
strncat(exp22,exp0,i);
strrev(exp0); exp1[0]='\0';
strncat(exp1,exp0,l-(i+1)); strrev(exp1);  printf("Three address
code:\ntemp=%s\n%s=temp\n",exp1,exp22);
break; case 2: printf("\nEnter the expression with arithmetic
operator:"); scanf("%s",ex); strcpy(exp0,ex); l=strlen(exp0);
exp1[0]='\0'; for(i=0;i<l;i++){ if(exp0[i]=='+'||exp0[i]=='-'){
if(exp0[i+2]=='/'||exp0[i+2]=='*'){ pm(); break;} else{ plus(); break;}
}
else if(exp0[i]=='/'||exp0[i]=='*'){
divi(); break;} } break; case 3:  printf("Enter the
expression with relational operator");
scanf("%s%s%s",id1,op,id2);
if(((strcmp(op,"<")==0)||(strcmp(op,"&gt;")==0)||(strcmp(op,"<=")==0)||(strcmp(op,"&gt;=")
==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(exp0);
j=l-i-1;
strncat(exp1,exp0,j);
strrev(exp1);
printf("Three address code:\ntemp=%s\ntemp1=%c%ctemp\n",exp1,exp0[j+1],exp0[j]);
} void divi(){ strncat(exp1,exp0,i+2);
printf("Three address code:\ntemp=%s\ntemp1=temp%c%c\n",exp1,exp0[i+2],exp0[i+3]);
} void plus(){ strncat(exp1,exp0,i+2);
printf("Three address code:\ntemp=%s\ntemp1=temp%c%c\n",exp1,exp0[i+2],exp0[i+3]); }
```