Sample Input and Output:

Enter Arithmetic Expression: a=b+c*d

THREE ADDRESS CODE

t1 = c * d

t2 = b + t1

a = t2

Code:

tcode.l

```
%{
#include "y.tab.h"
%}
%%
[0-9]+? {yylval.sym=(char)yytext[0]; return NUMBER;}
[a-zA-Z]+? {yylval.sym=(char)yytext[0];return LETTER;}
\n {return 0;}
. {return yytext[0];}
%%
int yywrap()
{
return 0;
}
```

YACC Specification:

tcode.y

```
%{
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
void ThreeAddressCode();
char AddToTable(char ,char, char);
int ind=0;//count number of lines
char temp = '1'; //for t1,t2,t3.....
struct incod
{
char opd1;
char opd2;
char opr;
```

```
};
%}
%union
{
char sym;
}
%token <sym> LETTER NUMBER
%type <sym> expr
%left '+'
%left '*''/'
%left '-'
%%
statement: LETTER '=' expr ';' {AddToTable((char)$1,(char)$3,'=');}
| expr ';'
;
expr:
expr '+' expr {$$ = AddToTable((char)$1,(char)$3,'+');}
| expr '-' expr {$$ = AddToTable((char)$1,(char)$3,'-');}
| expr '*' expr {$$ = AddToTable((char)$1,(char)$3,'*');}
| expr '/' expr {$$ = AddToTable((char)$1,(char)$3,'/');}
| '(' expr ')' {$$ = (char)$2;}
| NUMBER {$$ = (char)$1;}
| LETTER {$$ = (char)$1;}
|'-' expr {$$ = AddToTable((char)$2,(char)'\t','-');}
;

%%
yyerror(char *s)
{
printf("%s",s);
exit(0);
}
struct incod code[20];
```

Dept. Vision: To produce globally competent, innovative and socially responsible computing professionals

```
char AddToTable(char opd1,char opd2,char opr)
{
```

```
code[ind].opd1=opd1;
code[ind].opd2=opd2;
code[ind].opr=opr;
ind++;
return temp++;
}
void ThreeAddressCode()
{
int cnt = 0;
char temp = '1';
printf("\n\n\t THREE ADDRESS CODE\n\n");

while(cnt<ind)
{
if(code[cnt].opr != '=')
printf("t%c : = \t",temp++);
if(isalpha(code[cnt].opd1))
printf(" %c\t",code[cnt].opd1);
else if(code[cnt].opd1 >='1' && code[cnt].opd1 <='9')
printf("t%c\t",code[cnt].opd1);
printf(" %c\t",code[cnt].opr);
if(isalpha(code[cnt].opd2))
printf(" %c\n",code[cnt].opd2);
else if(code[cnt].opd2 >='1' && code[cnt].opd2 <='9')
printf("t%c\n",code[cnt].opd2);
cnt++;
}
}

main()
{
printf("\n Enter the Expression : ");
yyparse();
ThreeAddressCode();
}
```

OUTPUT:

```
C:\Users\Administrator\Desktop>flex tcode.l

C:\Users\Administrator\Desktop>bison -dy tcode.y

C:\Users\Administrator\Desktop>gcc lex.yy.c y.tab.c

C:\Users\Administrator\Desktop>a.exe

Enter the Expression : a=b+c*d;


         THREE ADDRESS CODE

t1 : =   c        *       d
t2 : =   b        +      t1
 a       =       t2
```

Problem understanding and algorithm (10)	Implementation (30)	Viva (10)	Total (50)

Result:

Thus the three address code for a simple program using LEX and YACC has been generated and verified.