NAME: VAIBHAV BANKA

REG NO: 21BCE1955

EXP 9

1. Write a yacc program to convert infix to prefix.

Lex Code

```
%{

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "y.tab.h"

%}

NUM [0-9]+

CHARACTER [_a-zA-Z][_a-zA-Z0-9]*

%%

exit.* { return EXIT; }

quit.* { return EXIT; }

{NUM} {yylval.exp = strdup(yytext); return NUM;}

{CHARACTER} { yylval.exp = strdup(yytext); return CHARACTER;}

[+-] { yylval.exp = strdup(yytext); return OPR1; }

[/*] { yylval.exp = strdup(yytext); return OPR2; }

[()] { return yytext[0]; }

\n { return NEWLINE; }

. ;

%%
```

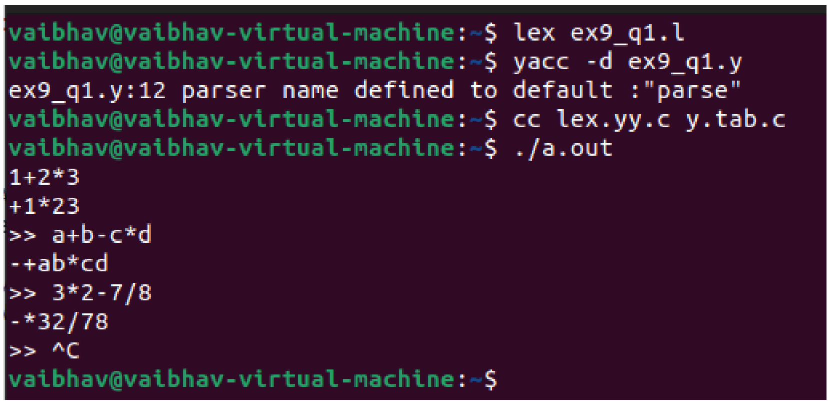
YACC program

```
%{

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

extern int yylex();
```

```
int yyerror(const char *p);
char *concat(const char* s1, const char* s2, const char*s3);
%}
%union {
char *exp;
int val;
};
%token NUM CHARACTER OPR1 OPR2 NEWLINE EXIT
%left OPR1 %left OPR2 %start lines
%%
lines: /*empty*/
|lines exp NEWLINE { printf("%s\n>> ",$<exp>2);}
;
exp: exp OPR1 exp {$<exp>$ = concat($<exp>2,$<exp>1,$<exp>3);}
|exp OPR2 exp {$<exp>$ = concat($<exp>2,$<exp>1,$<exp>3);}
|'(' exp ')' {$<exp>$ = $<exp>2;}
|NUM{$<exp>$ = $<exp>1;}
|CHARACTER {$<exp>$ = $<exp>1;}
|EXIT {exit(0);}
;
%%
int yywrap(){
return 1;
}
int main(){
yyparse();
}
char *concat(const char* s1, const char* s2, const char*s3){
int len = strlen(s1) + strlen(s2) + strlen(s3) + 1;
char *s = malloc(sizeof(char)*len);
int i=0;
```

```
for(int j=0; s1[j]!='\0'; j++)

s[i++] = s1[j];

for(int j=0; s2[j]!='\0'; j++)

s[i++] = s2[j];

for(int j=0; s3[j]!='\0'; j++)

s[i++] = s3[j];

s[i] = '\0';

return s;}

int yyerror(const char *p){

printf("%s\n",p); return 1;

}
```

OUTPUT



```
vaibhav@vaibhav-virtual-machine:~$ lex ex9_q1.l
vaibhav@vaibhav-virtual-machine:~$ yacc -d ex9_q1.y
ex9_q1.y:12 parser name defined to default :"parse"
vaibhav@vaibhav-virtual-machine:~$ cc lex.yy.c y.tab.c
vaibhav@vaibhav-virtual-machine:~$ ./a.out
1+2*3
+1*23
>> a+b-c*d
-+ab*cd
>> 3*2-7/8
-*32/78
>> ^C
vaibhav@vaibhav-virtual-machine:~$
```

2. Write yacc program to generate 3 address code for arithmetic expression.

LEX

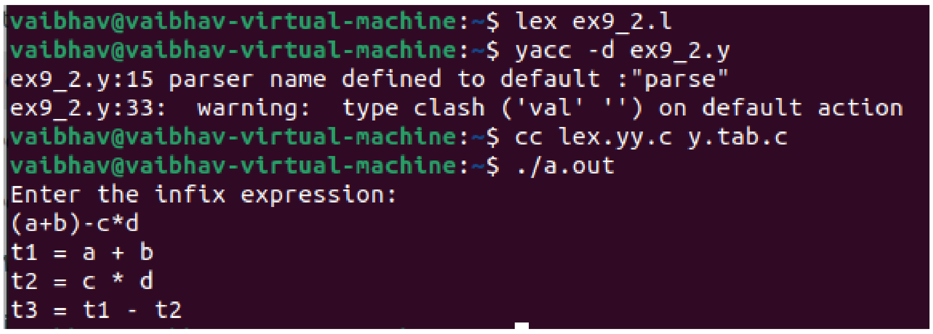
```
%{
 #include<string.h>
 #include "y.tab.h"
%}
%%
[a-z] {yylval.val=yytext;return NUM;}
[\-\n] {return *yytext;}
. {return yytext[0];}
%%
int yywrap(){}
```

YACC

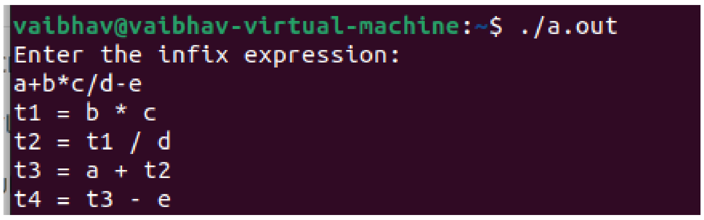
```
%{
 #include<stdio.h>
 #include<string.h>
 char temp[3]="t1";
 char st[10][10];
 int top=-1;
 int yylex(void);
 int yyerror(char *s);
 void codegen(char);
 void push(char*);
%}
%union
{
 char *val;
}
%token<val>NUM
%type<val>E
```

```
%type<val>T
%left '+''-'
%left '*''/'
%left '('')'
%%
S: E {return 0;}
;
E: E '+' T {codegen('+');}
   | E '-' T {codegen('-');}
   | T
;
T: T '*' F {codegen('*');}
   | T '/' F {codegen('/');}
   | G
;
F: G'^'F {codegen('^');}
   |G
;
G: '('E')'
   |H
;
H: NUM {push($1);}
;
%%
int main()
{
  printf("Enter the infix expression:\n");
  yyparse();
  return 0;
}
int yyerror(char* s){
  printf("\n Expression is invalid\n");
}
void push(char *ch)
{
  top=top+1;
  strcpy(st[top],ch);
}
void codegen(char a)
{
  printf("%s = %s %c %s\n",temp,st[top-1],a,st[top]);
  top-=1;
  strcpy(st[top],temp);
  temp[1]++;
}
```

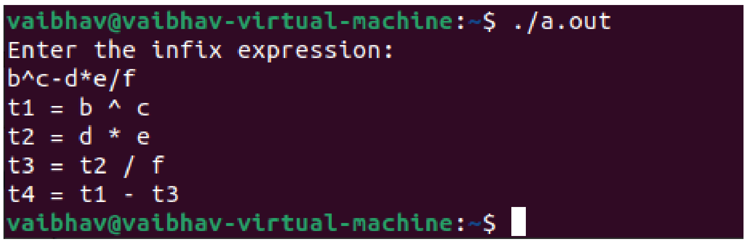
OUTPUT



```
vaibhav@vaibhav-virtual-machine:~$ lex ex9_2.l
vaibhav@vaibhav-virtual-machine:~$ yacc -d ex9_2.y
ex9_2.y:15 parser name defined to default :"parse"
ex9_2.y:33:  warning:  type clash ('val' '') on default action
vaibhav@vaibhav-virtual-machine:~$ cc lex.yy.c y.tab.c
vaibhav@vaibhav-virtual-machine:~$ ./a.out
Enter the infix expression:
(a+b)-c*d
t1 = a + b
t2 = c * d
t3 = t1 - t2
```



```
vaibhav@vaibhav-virtual-machine:~$ ./a.out
Enter the infix expression:
a+b*c/d-e
t1 = b * c
t2 = t1 / d
t3 = a + t2
t4 = t3 - e
```



```
vaibhav@vaibhav-virtual-machine:~$ ./a.out
Enter the infix expression:
b^c-d*e/f
t1 = b ^ c
t2 = d * e
t3 = t2 / f
t4 = t1 - t3
vaibhav@vaibhav-virtual-machine:~$
```

3. Write yacc program to generate 3 address code for while loop.

LEX

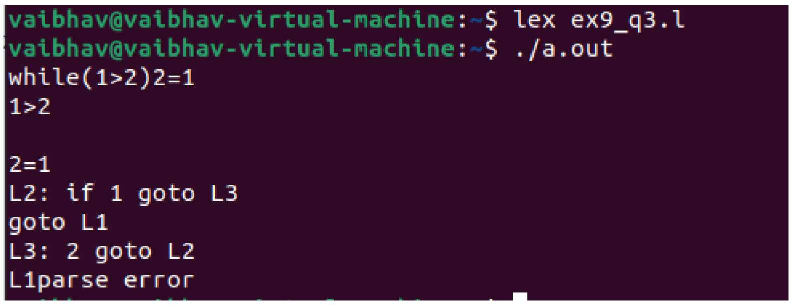
```
%{
#include "y.tab.h"
%}

%%
while      { return WHILE; }
[0-9]+     { yylval= atoi(yytext); return NUMBER; }
"+"        { return PLUS; }
"-"        { return MINUS; }
"*"        { return MULTIPLY; }
"/"        { return DIVIDE; }
"="        { return EQUAL; }
"<"        { return LESSTHAN; }
">"        { return GREATERTHAN; }
"&&"       { return AND; }
"||"       { return OR; }
"("        { return LPAREN; }
")"        { return RPAREN; }
";"        { return SEMICOLON; }
\n         { /* ignore newlines */ }
[ \t]      { /* ignore whitespace */ }
.          { printf("Invalid character %c\n", yytext[0]); }
%%

int yywrap(){}
```

YACC

```
%{
 #include <stdio.h>
 int label_count = 1;
 int yylex(void);

 int yyerror(char *s);

%}


%token WHILE NUMBER PLUS MINUS MULTIPLY DIVIDE EQUAL LESSTHAN GREATERTHAN AND OR
LPAREN RPAREN SEMICOLON

%left OR

%left AND

%left GREATERTHAN GREATERTHANOREQUAL LESSTHAN LESSTHANOREQUAL

%left PLUS MINUS

%left MULTIPLY DIVIDE


%%


program : statement

;

statement : assignment

        | while_statement

;

assignment : NUMBER EQUAL expression

;

expression : NUMBER

        | expression PLUS expression {printf("%d+%d",$1,$3);}

        | expression MINUS expression {printf("%d-%d",$1,$3);}

        | expression MULTIPLY expression {printf("%d*%d",$1,$3);}

        | expression DIVIDE expression {printf("%d/%d",$1,$3);}

        | expression EQUAL expression {printf("%d=%d",$1,$3);}

        | expression LESSTHAN expression {printf("%d<%d",$1,$3);}

        | expression GREATERTHAN expression {printf("%d>%d",$1,$3);}

        | LPAREN expression RPAREN

;
```

```
while_statement : WHILE LPAREN expression RPAREN statement
                {
                  printf("L%d: ", label_count+1); // start of loop label
                  printf("if %d goto L%d\n", $3, label_count+2);
                  printf("goto L%d\n", label_count);
                  printf("L%d: ", label_count+2); // end of loop label
                  printf("%d ", $5);
                  printf("goto L%d \n",label_count+1);
                  printf("L%d",label_count);
                }
              ;

%%
int main()
{
   yyparse();
   return 0;
}

int yyerror(char *s)
{
   printf("%s\n", s);
}
```

OUTPUT



```
vaibhav@vaibhav-virtual-machine:~$ lex ex9_q3.l
vaibhav@vaibhav-virtual-machine:~$ ./a.out
while(1>2)2=1
1>2

2=1
L2: if 1 goto L3
goto L1
L3: 2 goto L2
L1parse error
```