// NAME: AHAN BANDYOPADHYAY

//ROLL No. 211210008

//Compiler Design Lab 10

1. Using YACC, write semantic actions to check if the parenthesis are balanced and count the number of matching parenthesis

P->(P) | a

```
%{
#include <stdio.h>
int parenthesis_count = 0;
int is_balanced = 1;
%}
%token LPAREN RPAREN A
%start P
%%
P : LPAREN P RPAREN { parenthesis_count++; } | A;
A : 'a' ;
LPAREN : '(';
RPAREN : ')' ;
%%
int main() {
 yyparse();
   if (is_balanced && parenthesis_count == 0) {
      printf("Parentheses are balanced and there are no unmatched parentheses.\n");
   } else {
      printf("Parentheses are not balanced or there are unmatched parentheses.\n");
   }
   return 0;
}
int yyerror(const char *s) {
   printf("Error: %s\n", s);
   return 0;
}
```

Input: "(a)"

Output: Parentheses are balanced and there are no unmatched parentheses.

Input: "(a"

Output: Parentheses are not balanced or there are unmatched parentheses.

2. Using YACC, write semantic actions that translate arithmetic expressions (generated from the given grammar) from infix into postfix notation

E -> E + T E -> T

T-> T * F T -> F

F -> (E) F -> num

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

%token NUM
%token PLUS MINUS TIMES DIVIDE LPAREN RPAREN
%left PLUS MINUS
%left TIMES DIVIDE
%start E

%%

E : E PLUS T { printf("%s ", $1); printf("%s ", $3); printf("+ "); }| T;
T : T TIMES F { printf("%s ", $1); printf("%s ", $3); printf("* "); }| F;
F : LPAREN E RPAREN { printf("("); printf("%s ", $2); printf(") "); }
 | NUM { printf("%s ", $1); };
NUM : [0-9]+
%%

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

int yylex() {
```

```
    int c = getchar();
    if (isdigit(c)) {
        ungetc(c, stdin);
        int num;
        scanf("%d", &num);
        return NUM;
    } else if (c == '+') {
        return PLUS;
    } else if (c == '-') {
        return MINUS;
    } else if (c == '*') {
        return TIMES;
    } else if (c == '/') {
        return DIVIDE;
    } else if (c == '(') {
        return LPAREN;
    } else if (c == ')') {
        return RPAREN;
    } else if (c == EOF) {
        return 0;
    } else {
        return -1; // Error
    }
}

int yyerror(const char *s) {
    printf("Error: %s\n", s);
    return 0;
}
```