Title: Generating abstract syntax tree using LEX and YACC.

LEX File:

```
%{
#include "y.tab.h"
%}
%%
[0-9]+   {yylval = (int)yytext; return NUMBER;}
[\t\n]   ;
"+"      return(PLUS);
"-"      return(MINUS);
"*"      return(TIMES);
"/"      return(DIVIDE);
"^"      return(POWER);
"("      return(LEFT_PARENTHESIS);
")"      return(RIGHT_PARENTHESIS);
";"      return(END);
%%
int yywrap (void)
{return 1;}
```

YACC File:

```
%{
#include <stdio.h>
  typedef struct node
  {
    struct node *left;
    struct node *right;
    char *token;
  } node;
  node *mknode(node *left, node *right, char *token);
  void printtree(node *tree);

#define YYSTYPE struct node *
%}

%start lines
%token     NUMBER
%token     PLUS  MINUS TIMES DIVIDE     POWER
%token     LEFT_PARENTHESIS RIGHT_PARENTHESIS
%token     END
%left PLUS  MINUS
%left TIMES DIVIDE
%right     POWER
%%
lines:  /* empty */
        | lines line /* do nothing */

line:   exp END       { printtree($1); printf("\n");}
     ;

exp    : term             {$$ = $1;}
        | exp PLUS term     {$$ = mknode($1, $3, "+");}
        | exp MINUS term    {$$ = mknode($1, $3, "-");}
```

```
        ;

term    : factor           {$$ = $1;}
        | term TIMES factor  {$$ = mknode($1, $3, "*");}
        ;

factor : NUMBER            {$$ = mknode(0,0,(char *)yylval);}
        | LEFT_PARENTHESIS exp RIGHT_PARENTHESIS {$$ = $2;}
        ;
%%
int main (void) {return yyparse ( );}

node *mknode(node *left, node *right, char *token)
{ /* malloc the node */
  node *newnode = (node *)malloc(sizeof(node));
  char *newstr = (char *)malloc(strlen(token)+1);
  strcpy(newstr, token);
  newnode->left = left;
  newnode->right = right;
  newnode->token = newstr;
  return(newnode);
}

void printtree(node *tree)
{
  int i;
  if (tree->left || tree->right)
    printf("(");

  printf(" %s ", tree->token);

  if (tree->left)
    printtree(tree->left);
  if (tree->right)
    printtree(tree->right);

  if (tree->left || tree->right)
    printf(")");
}
int yyerror (char *s) {fprintf (stderr, "%s\n", s);}
```

Output:

```
pvg@pvg:~/Desktop/ast$ yacc -d ast.y
pvg@pvg:~/Desktop/ast$ lex ast.l
pvg@pvg:~/Desktop/ast$ gcc lex.yy.c y.tab.c
pvg@pvg:~/Desktop/ast$ ./a.out
2*(3+5)*6;
( * ( *  2 ( +  3  5 )) 6 )
```