EXAMPLE_FILE:

```
Read >> A
Read >> B
while #B != 0:
    R = A % B
    A = B
    B = R
Write << B
```

LEX_FILE:

```
%{
    #include <math.h>
    #include<string.h>
#include <stdlib.h>
#include <stdio.h>

int next_pos_PIF = 0;
char PIF[1000];
char ST[25][25];
int ST_size = 1;
FILE* PIF_descr;
FILE* ST_descr;

int adaugare_in_ST(char token[]) {
    int i = 0;
    for(i; i<ST_size; i++){
        if(strcmp(ST[i], token) == 0) {
            return i;
        }
    }

    ST_descr = fopen("ST.out","w");
    strcpy(ST[ST_size++], token);
    for(i=0; i<ST_size; i++){
        fprintf(ST_descr, "%d %s\n", i, ST[i]);
    }
    fclose(ST_descr);
    return ST_size-1;
}

char* itoa2(int val, int base){
        static char buffer[32] = {0};
    char zero_str[2];
    zero_str[0] = '0';
    zero_str[1] = 0;
    char* zero_ptr = zero_str;
    if(val == 0)
        return zero_ptr;
```

```
        int i = 30;

        for(; val && i ; --i, val /= base)

                buffer[i] = "0123456789abcdef"[val % base];

        return &buffer[i+1];

}


void adaugare_in_PIF(char token[], int position) {
    strcpy(PIF+next_pos_PIF, token);
    next_pos_PIF+=strlen(yytext);
    PIF[next_pos_PIF++] = '=';
    char pos_str[5];
    strcpy(pos_str,itoa2(position, 10));

    strcpy(PIF+next_pos_PIF, pos_str);
    next_pos_PIF += strlen(pos_str);
    PIF[next_pos_PIF++] = ' ';
    PIF_descr = fopen("PIF.out","w");
    fprintf(PIF_descr, "%s \r\n\r ", PIF);
    fclose(PIF_descr);
}
%}
DIGIT [0-9]
LETTER [a-z][A-Z]
ID              [a-zA-Z][a-zA-Z0-9]*
%option noyywrap
%%
"{"[^}\n]*"}"           /* eat up one-line comments */
[ \t\n]+                /* eat up whitespace */
[+-]?{DIGIT}+   {printf( "An integer: %s (%d)\n", yytext, atoi( yytext ) );  }
"Read" { adaugare_in_PIF(yytext, 100); printf( "A keyword: %s\n", yytext );  }
"Write" { adaugare_in_PIF(yytext, 100); printf( "A keyword: %s\n", yytext );  }
"var" { adaugare_in_PIF(yytext, 100); printf( "A keyword: %s\n", yytext );  }
"if" { adaugare_in_PIF(yytext, 100); printf( "A keyword: %s\n", yytext );  }
"else" { adaugare_in_PIF(yytext, 100); printf( "A keyword: %s\n", yytext );  }
"not" { adaugare_in_PIF(yytext, 100); printf( "A keyword: %s\n", yytext );   }
"end" { adaugare_in_PIF(yytext, 100); printf( "A keyword: %s\n", yytext );  }
"and" { adaugare_in_PIF(yytext, 100); printf( "A keyword: %s\n", yytext );  }
"or" { adaugare_in_PIF(yytext, 100); printf( "A keyword: %s\n", yytext );  }
"xor" { adaugare_in_PIF(yytext, 100); printf( "A keyword: %s\n", yytext );  }
"while" { adaugare_in_PIF(yytext, 100); printf( "A keyword: %s\n", yytext );  }
"exit" { adaugare_in_PIF(yytext, 100); printf( "A keyword: %s\n", yytext );  }
'.' { adaugare_in_PIF(yytext, 100); printf( "A character: %s\n", yytext ); }
\".*\" { adaugare_in_PIF(yytext, 100); printf( "A string: %s\n", yytext );  }
"!=" { adaugare_in_PIF(yytext, 100); printf( "A separator: %s\n", yytext );  }
"==" { adaugare_in_PIF(yytext, 100); printf( "A separator: %s\n", yytext );  }
```

```
">=" { adaugare_in_PIF(yytext, 100); printf( "A separator: %s\n", yytext );  }
">>" { adaugare_in_PIF(yytext, 100); printf( "A separator: %s\n", yytext );  }
"<<" { adaugare_in_PIF(yytext, 100); printf( "A separator: %s\n", yytext );  }
"<=" { adaugare_in_PIF(yytext, 100); printf( "A separator: %s\n", yytext );  }
"+" { adaugare_in_PIF(yytext, 100); printf( "A separator: %s\n", yytext );  }
"-" { adaugare_in_PIF(yytext, 100); printf( "A separator: %s\n", yytext );  }
"*" { adaugare_in_PIF(yytext, 100); printf( "A separator: %s\n", yytext );  }
"/" { adaugare_in_PIF(yytext, 100); printf( "A separator: %s\n", yytext );  }
"%" { adaugare_in_PIF(yytext, 100); printf( "A separator: %s\n", yytext );  }
"<" { adaugare_in_PIF(yytext, 100); printf( "A separator: %s\n", yytext );  }
"=" { adaugare_in_PIF(yytext, 100); printf( "A separator: %s\n", yytext );  }
">" { adaugare_in_PIF(yytext, 100); printf( "A separator: %s\n", yytext );  }
"[" { adaugare_in_PIF(yytext, 100); printf( "A separator: %s\n", yytext );  }
"]" { adaugare_in_PIF(yytext, 100); printf( "A separator: %s\n", yytext );  }
"{" { adaugare_in_PIF(yytext, 100); printf( "A separator: %s\n", yytext );  }
"}" { adaugare_in_PIF(yytext, 100); printf( "A separator: %s\n", yytext );  }
"(" { adaugare_in_PIF(yytext, 100);printf( "A separator: %s\n", yytext );  }
")"     { adaugare_in_PIF(yytext, 100);printf( "A separator: %s\n", yytext );  }
":" { adaugare_in_PIF(yytext, 100);printf( "A separator: %s\n", yytext );  }
";" { adaugare_in_PIF(yytext, 100);printf( "A separator: %s\n", yytext );  }
{ID}  {int position = adaugare_in_ST(yytext); adaugare_in_PIF(yytext, position);
printf( "An identifier: %s\n", yytext );  }
. printf("Eroare\n");
%%
main( argc, argv )
int argc;
char **argv;
{
    ++argv, --argc; /* skip over program name */
    if ( argc > 0 )
    yyin = fopen( argv[0], "r" );
    else
     yyin = stdin;
    yylex();
}
```