```
%{
#include <ctype.h>
#include <stdio.h>
char lexema[255];
void yyerror(char *);
int yylex();
%}

%token ID NUM

%%
expr: ID ':' '=' term expr
    | %empty;
term: term '+' NUM
    | term '-' NUM
    | NUM;
%%

void yyerror(char *mgs)
{
  printf("error: %s",mgs);
}

int yylex()
{
  char c;

  while(1) {
    c = getchar();

    if(c == '\n') continue;
    if(c == ' ') continue;
    if(isspace(c)) continue;

    if(isalpha(c)) {
      int i = 0;
      do{
        lexema[i++] = c;
        c = getchar();
      } while(isalnum(c));
      ungetc(c, stdin);
      lexema[i] = 0;
      return ID;
    }

    if(isdigit(c)) {
```

```
      int i = 0;
      do{
        lexema[i++] = c;
        c = getchar();
      } while(isdigit(c));
      ungetc(c, stdin);
      lexema[i] = 0;
      return NUM;
    }

    return c;
  }
}

int main()
{
  if(!yyparse()) printf("cadena valida\n");
  else printf("cadena invalida\n");
  return 0;
}
```

```
[wensespl@LAPTOP-1D2RDUOO ···/USUARIO/Desktop/Compiladores/semana8]

$ ./a.out
x:=2+3-5
cadena valida
```

Estados	Cadena
	X:=2+3-5
0 (ID shift, and go to state 1)	ID ‘:’ ‘=’ NUM ‘+’ NUM ‘-’ NUM
01 (':' shift, and go to state 3)	‘:’ ‘=’ NUM ‘+’ NUM ‘-’ NUM
013 ('=' shift, and go to state 5)	‘=’ NUM ‘+’ NUM ‘-’ NUM
0135 (NUM shift, and go to state 6)	NUM ‘+’ NUM ‘-’ NUM
01356 (reduce using rule 5 (term))	‘+’ NUM ‘-’ NUM
0135 (term go to state 7)	‘+’ NUM ‘-’ NUM
01357 ('+' shift, and go to state 8)	‘+’ NUM ‘-’ NUM
013578 (NUM shift, and go to state 11)	NUM ‘-’ NUM
013578 11 (reduce using rule 3 (term))	‘-’ NUM
0135 (term go to state 7)	‘-’ NUM
01357 ('-' shift, and go to state 9)	‘-’ NUM
013579 (NUM shift, and go to state 12)	NUM
013579 12 (reduce using rule 4 (term))	
0135 (term go to state 7)	
01357 (reduce using rule 2 (expr)) (expr go to state 10)	
01357 10 (reduce using rule 1 (expr))	
0 (expr go to state 2)	
02 (shift, and go to state 4)	
024	accept