Exercício 0.
Objetivo: Localizar TODAS as regras da sua Gramática que declaram ou utilizam identificador.
A fazer: Informar o projetista do compilador de toda ocorrência de declaração ou uso de identificador (de forma genérica)
Dica: Usar printf

Exercício 1.
Objetivo: Nas regras que você localizou acima, dar mensagens personalizadas, ou seja, informar o programador QUAL identificador foi localizado sendo declarado ou utilizado.
Dica: Usar, além do printf, $n. Para ter acesso ao identificador como string, lembrar de afetar a variável da pilha yylval.

Colocar o trecho abaixo antes de suas declarações de tokens no seu .y
%union { char *cadeia; }
e colocar o trecho seguinte no .l, na ação semântica correspondente à Expressão Regular que define seu identificador (antes do return ID):
yylval.cadeia= (char *) strdup(yytext);

Exercício 2.
Objetivo: Fazer tratamento de ERRO e WARNING para identificadores de variáveis e de procedimentos.
ERRO: Quando houver ao menos uma variável ou procedimento USADO que não tenha sido declarado.
WARNING: Quando houver ao menos uma variável ou procedimento DECLARADO que não esteja sendo usado no programa.
A fazer: Montar a estrutura de dados correspondente à Tabela de Símbolos e duas funções de manutenção (insere e busca).

Obs.: A estrutura de dados é uma lista encadeada que implementa a Tabela de Símbolos.
Por exemplo, a seguinte tabela será gerada após varredura do programa abaixo:

NOME	TIPO	USADA
i	inteiro	sim
j	inteiro	nao
var1	flutuante	sim

```
var
        i,j: inteiro;
        var1: flutuante;
{
        i:=2;
        var1:=2.5;
}
```

O seu compilador deverá informar ao programador se o programa está semanticamente correto ou não (se há erro ou não) e se há warning.