


5COP093 - Lista de Exercícios 15

1.  **Exercício Prático:** Considere a gramática a seguir:

```
S → D D' $  
D' → D D'  
D' →  
D → T I  
T → int  
T → float  
T → char  
I → id  
I → I , id
```

O *token* `id` é dado pela seguinte expressão regular: `[a-z]+`

Utilizando as ferramentas `flex` e `bison`, implemente um programa que armazena em uma tabela de símbolos os identificadores e os seus tipos. Além disso o seu programa deve verificar se um determinado identificador já existe e, em caso positivo, emitir a mensagem de erro adequada.

Para ilustrar as mensagens de erro, considere os exemplos apresentados a seguir:

Entrada: `int a, b, c char a $`

Saída: `redefinition of identifier 'a'`

Entrada: `int a, b, cc char c int b $`

Saída: `identifier 'b' already declared`

Entrada: `int a, b, c char a int b $`

Saída: `redefinition of identifier 'a'
identifier 'b' already declared`

A mensagem `redefinition of identifier '<nome-do-identificador>'` deve ser mostrada quando já existir na tabela de símbolos um identificador de mesmo nome, mas tipo diferente.

A mensagem `identifier '<nome-do-identificador>' already declared` deve ser mostrada quando já existir na tabela de símbolos um identificador de mesmo nome e mesmo tipo.

Observação: Somente a primeira ocorrência de um dado identificador é armazenada na tabela de símbolos. As demais ocorrências de um identificador já existente na tabela de símbolos irão gerar mensagens de erro.

As cadeias de entrada para o seu programa estarão armazenadas em um arquivo, sendo que haverá uma cadeia por linha. Espaços em branco e quebras de linha devem ser removidas pelo analisador léxico sem acusar erro. Os arquivos de entrada estarão corretos do ponto de vista léxico e sintático, isto é, tais análises não devem encontrar nenhum erro.

O arquivo contendo as cadeias deve ser lido da entrada padrão e a saída do programa deve ser impressa na saída padrão. Supondo que o arquivo de entrada se chame `cadeias.txt` e que o programa executável se chame `l15e1`, a execução será da seguinte forma:

```
./l15e1 < cadeias.txt
```

Se uma determinada cadeia de entrada não gerar nenhuma mensagem de erro, o seu programa deve mostrar a seguinte mensagem:

```
All Identifiers on Hash.
```

Para cada mensagem que o seu programa gerar, ele deve indicar a linha que gerou a mesma. Como exemplo completo de entrada e saída, considere o exemplo de entrada mostrado a seguir, o qual contém 3 linhas:

Entrada:

```
int a, b, c char a $  
int a, b, c char c int b $  
int a, b, c char d int e $
```

Saída Esperada:

```
1: redefinition of identifier 'a'  
2: redefinition of identifier 'c'  
2: identifier 'b' already declared  
3: All Identifiers on Hash.
```

De modo a testar o seu programa, utilize as entradas e saídas fornecidas. Utilize o programa `diff` para comparar a sua saída com a saída esperada. Sua saída só pode ser considerada correta quando estiver idêntica à saída esperada.