MC202 — ESTRUTURAS DE DADOS

Laboratório 13 — Hashing

Tarefa

A publicação de artigos é a principal ferramenta de divulgação científica. Se dois pesquisadores assinam um mesmo artigo, dizemos que eles são colaboradores. A sua tarefa é escrever um programa que, a partir de uma lista de autores de artigos, determine se existe colaboração entre dois autores.

Neste laboratório a tabela deverá utilizar endereçamento aberto e é obrigatório o uso de **hashing duplo.** O tamanho da tabela não pode ser maior que 2^{11} (há no máximo duas mil colaborações distintas).

Entrada

A primeira linha contém dois inteiros positivos n e m que indicam respectivamente o número de artigos e o número de consultas. Cada uma das n linhas seguintes contém a lista de autores de um artigo com autores separados por vírgulas. A lista termina com um ponto (veja o exemplo abaixo). Cada autor é identificado pela inicial de seu nome, seguida de um ponto e de um espaço em branco, seguida de seu último sobrenome. O sobrenome de um autor possui no máximo 15 caracteres. As próximas m linhas contêm pares de autores.

Saída

O seu programa deve produzir m linhas na saída. Cada linha deve conter um único caractere, sendo: S se existe colaboração entre os autores, ou N caso contrário.

Exemplo

Entrada

```
13 5
R. Sedgewick.
T. Cormen, C. Leiserson, R. Rivest, C. Stein.
A. Aho, J. Hopcroft, J. Ullmann.
W. Celes, R. Cerqueira, J. Rangel.
M. Folk, B. Zoellick.
F. Lorenzi, P. Mattos, T. Carvalho.
S. Pereira.
E. Reingold, W. Hanson.
J. Szwarcfiter, L. Markenzon.
```

```
D. Knuth.
N. Wirth.
A. Tenembaum.
N. Ziviani.
J. Hopcroft, J. Ullmann.
T. Cormen, S. Pereira.
B. Zoellick, M. Folk.
F. Lorenzi, T. Carvalho.
W. Celes, D. Knuth.
```

Saída

```
S
N
S
S
N
```

Dicas

- Você pode ler o nome de um autor usando:
  `char inicial, sobrenome[16], separador;`
  `scanf(" %c. %[^.,]%c", &inicial, sobrenome, &separador);`
  que lê em sequência: um caractere, uma string (sem ponto ou vírgula) e um caractere (que pode ser um ponto ou uma vírgula).

Critérios específicos

- Para as turmas E e F, este laboratório tem peso 5.
- Para as turmas G e H, este laboratório tem peso 2.
- Deverão ser submetidos os seguintes arquivos: **lab13.c**, **hash.h** e **hash.c**.
- Tempo máximo de execução: 1 segundo.

Testando

Para compilar com o Makefile fornecido e verificar se a solução está correta basta seguir o exemplo abaixo.

```
make
./lab13 < arq01.in > arq01.out
diff arq01.out arq01.res
```

onde `arq01.in` é a entrada (casos de testes disponíveis no SuSy) e `arq01.out` é a saída do seu programa. O Makefile também contém uma regra para testar todos os testes de uma vez; nesse caso, basta digitar:

```
make testar_tudo
```

Observações gerais

No SuSy, haverá 3 tipos de tarefas com siglas diferentes para cada laboratório de programação. Todas possuirão os mesmos casos de teste. As siglas são:

1. **SANDBOX:** Esta tarefa serve para testar o programa no SuSy antes de submeter a versão final. Nessa tarefa, tanto o prazo quanto o número de submissões são ilimitados, porém arquivos submetidos aqui **não serão corrigidos**.

2. **ENTREGA:** Esta tarefa tem limite de **uma única submissão** e serve para entregar a versão final dentro do prazo estabelecido para o laboratório. Não use essa tarefa para testar o seu programa: submeta aqui quando não for mais fazer alterações no seu programa.

3. **FORAPRAZO:** Esta tarefa tem limite de **uma única submissão** e serve para entregar a versão final após o prazo estabelecido para o laboratório, mas com nota reduzida (conforme a ementa). O envio nesta tarefa irá substituir a nota obtida na tarefa ENTREGA apenas se o aluno tiver realizado as correções sugeridas no feedback ou caso não tenha enviado anteriormente em ENTREGA.

Observações sobre SuSy:

- Versão do GCC: C-ANSI 4.8.2 20140120 (Red Hat 4.8.2-15).
- Flags de compilação:
  `-ansi -Wall -pedantic-errors -Werror -g -lm`
- Utilize comentários do tipo `/* comentário */`;
  comentários do tipo `//` serão tratados como erros pelo SuSy.

Além das observações acima, esse laboratório será avaliado pelos critérios gerais:

- Indentação de código e outras boas práticas, tais como:
  - uso de comentários (apenas quando forem relevantes);
  - código simples e fácil de entender;
  - sem duplicidade (partes que fazem a mesma coisa).
- Organização do código:

    - tipos de dados criados pelo usuário e funções bem definidas e tão independentes quanto possível.

- Corretude do programa:
    - programa correto e implementado conforme solicitado no enunciado;
    - eficiência do algoritmo (complexidade de tempo ou de espaço);
    - inicialização de variáveis sempre que for necessário;
    - desalocar toda memória alocada dinamicamente durante a execução do programa;
    - realizar leitura ou escrita em blocos de memória não alocados;
    - duplicidade de código;
    - má utilização de recursos (overhead de processamento ou de memória);
    - dentre outros critérios caso necessários.