MC202E - ESTRUTURAS DE DADOS

Lab02: Bonificação Salarial

A universidade pretende aplicar uma nova política de bonificação salarial que atrela a remuneração dos professores em função do desempenho dos alunos. A política de bonificação adota dará aumento salarial a um professor somente se **todos os seus alunos** obtiverem desempenho acadêmico de acordo com as seguintes regras:

- ❏ Se possuírem desempenho acadêmico igual a 10.0, então a bonificação consiste em um aumento de 15%;
- ❏ Caso contrário, se possuírem desempenho acadêmico igual ou maior que 9.0, então a bonificação consiste em um aumento de 10%;
- ❏ Caso contrário, se possuírem desempenho acadêmico igual ou maior que 8.5, então a bonificação consiste em um aumento de 5%.

Para ajudar na implementação da bonificação salarial dos professores, você deverá implementar um programa em C que aplica conceitos de abstração de dados, i.e., a utilização de **tipos abstratos de dados**.

Tarefa

Para esta tarefa você receberá o arquivo principal `lab02.c` e os arquivos `professor.*` e `aluno.*`, os quais estão disponíveis em **Arquivos auxiliares** no *SuSy* (compactados no arquivo Workspace.zip). Cada um dos **arquivos de cabeçalho** contém a declaração de uma estrutura e os protótipos de funções, os quais são necessários à compilação, e **não devem ser modificados e também não serão submetidos no *SuSy***. Detalhes de implementação são encontrados principalmente nos arquivos de cabeçalho.

Entrada

A primeira linha da entrada contém um inteiro positivo m que corresponde ao número de professores. Em seguida são dadas m linhas, com cada linha contendo o nome de um professor, o salário atual dele, e a disciplina que ele leciona. Cada professor leciona uma única disciplina que é identificada por um código alfanumérico de 5 caracteres. A próxima linha da entrada contém um inteiro positivo n que corresponde ao número de alunos. Cada uma das n linhas seguintes contém o registro acadêmico de um aluno, seu desempenho acadêmico, o número de disciplinas que o aluno está matriculado (no máximo 5), e a sua lista de disciplinas.

Restrições

- ❏ $3 \leq m \leq 100$
- ❏ $3 \leq n \leq 200$

Saída

A saída do seu programa apresentará m linhas, onde cada linha contém o nome do professor e o seu salário já contendo a bonificação salarial (caso o professor tenha recebido). Os professores serão listados na mesma ordem da entrada e o salário terá precisão de duas casas decimais.

Observação

❑ O código de leitura e saída já estão implementados e não devem ser modificados.

Exemplo

A grafia da saída abaixo deve ser seguida rigorosamente por seu programa, inclusive a impressão da linha em branco no final da saída.

Entrada

```
3
JoãoMoura 10000.00 MO202
MauroFernandes 9987.00 MO444
CarlosSousa 7988.00 MO501
6
2019001 10.00 2 MO444 MO202
2019002 8.50 1 MO202
2019003 8.00 1 MO501
2019004 10.00 2 MO501 MO444
2019005 9.50 1 MO501
2019006 9.75 3 MO444 MO202 MO501
```

Saída

```
JoãoMoura 10500.00
MauroFernandes 10985.70
CarlosSousa 7988.00

```

No exemplo acima, o professor `MauroFernandes` obteve uma bonificação de 10%, visto que todos os alunos da disciplina MO444 apresentaram desempenho acadêmico maior que 9.0; já o professor `JoãoMoura` obteve uma bonificação de 5% (por quê?); e o professor `CarlosSousa` não obteve uma bonificação (por quê?).

Critérios específicos

Os seguintes critérios específicos sobre o envio, implementação, compilação e execução devem ser satisfeitos.

i. Submeter no *SuSy* os arquivos:
- ⇒ `lab02.c`: programa principal, onde deverá ser implementado apenas a função `processar_aumento`. **Não é permitido** acessar diretamente os campos de um TAD na função a ser implementada.
- ⇒ `professor.c`: deverá conter a implementação correta das 5 funções (são 4 na verdade, pois uma delas já está implementada), conforme descrito no arquivo de cabeçalho do mesmo.
- ⇒ `aluno.c`: deverá conter a implementação correta das 4 funções, conforme descrito no arquivo de cabeçalho de cabeçalho do mesmo.

ii. É obrigatório seguir as especificações dos arquivos de cabeçalho.

iii. Flags de compilação:
```
-std=c99 -Wall -Werror -g -lm
```

iv. Tempo máximo de execução: 1 segundo.

Observações gerais

No decorrer do semestre haverá 3 tipos de tarefas no *SuSy* (descritas logo abaixo). As tarefas possuirão os mesmos casos de testes abertos e fechados, no entanto o número de submissões permitidas e prazos são diferentes. As seguintes tarefas estão disponíveis no *SuSy*:

- ❑ **Lab02-AmbienteDeTeste**: Esta tarefa serve para testar seu programa no *SuSy* antes de submeter a versão final. Nessa tarefa, tanto o prazo quanto o número de submissões são ilimitados, porém os arquivos submetidos aqui **não serão corrigidos**.
- ❑ **Lab02-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 e submeta aqui apenas quando não for mais fazer alterações no seu programa.
- ❑ **Lab02-ForaDoPrazo**: Esta tarefa tem limite de uma **única** submissão e serve para entregar a versão final fora prazo estabelecido para o laboratório. Esta tarefa irá substituir a nota obtida na tarefa **Lab02-Entrega** apenas se o aluno tiver realizado as correções sugeridas no *feedback* ou caso não tenha enviado anteriormente na tarefa **Lab02-Entrega**.

Avaliação

Este laboratório será avaliado da seguinte maneira: se o seu programa apresentar resposta correta para todos os casos de teste do *SuSy*, então é nota 10; caso contrário, é nota 0 (i.e, seu programa apresentou resposta incorreta para pelo menos um caso de teste aberto ou fechado).

Testando seu programa

Para compilar usando o `Makefile` fornecido e testar se a solução do seu programa está correta, basta seguir o exemplo abaixo.

```
make
./lab02 < testes_abertos/arq01.in > testes_abertos/arq01.out
diff testes_abertos/arq01.out testes_abertos/arq01.res
```

O `arq01.in` é a entrada (caso de teste disponível no *SuSy*) e `arq01.out` é a saída do seu programa. O `Makefile` também contém regras para baixar e testar todos os testes de uma única vez; nesse caso, basta digitar conforme o exemplo a seguir.

```
make baixar_abertos
```

```
make testar_abertos
```

A primeira instrução irá baixar os casos de teste abertos para a pasta **testes_abertos**, criada automaticamente, e a segunda instrução irá testar o seu programa com os casos de teste abertos. Após o prazo, os casos de teste fechados serão liberados e podem ser baixados e testados da mesma forma que os testes abertos. Para isso, basta trocar `"_abertos"` por `"_fechados"` (e.g., `make baixar_fechados`).