

# MC202 - Estruturas de Dados

## Lab 10

**Link da atividade:** [https://classroom.github.com/a/\\_uQ8Yrkw](https://classroom.github.com/a/_uQ8Yrkw)

**Peso:** 4

Warly, *the Culinarian*, é um exímio chef francês que também foi levado para *o Constant*. Ele é tão habilidoso na cozinha, que alguns de seus pratos têm propriedades extras (além de encher o bucho e curar onde dói, tipo a feijoada do bandejão). Isso significa que outros personagens podem pedir comidas específicas para situações mais ou menos urgentes.

Como gerenciar recursos perecíveis não é uma tarefa fácil, Warly assumiu totalmente a tarefa de preparar comida para os outros sobreviventes. O problema é que preparar condimentos e cozinhar vários pratos em paralelo já ocupa toda a atenção do nosso chef, sem deixar espaço para o gerenciamento da ordem dos pedidos.

Warly decidiu preparar os pratos em uma ordem mais elaborada do que uma fila normal (FIFO). A ideia é que quanto mais tempo um(a) sobrevivente passou no mundo, melhor é a relação dele(a) com Warly e maior é a prioridade dos pratos pedidos por ele(a). Além disso, imprevistos também modificam a prioridade de algum pedido. Por exemplo, se alguém estiver com o estômago vazio, um pedido novo com prioridade maior pode ser feito ou algum pedido que já foi feito anteriormente pode sofrer alteração de prioridade. Outro exemplo: vários sobreviventes podem pedir Fish Cordon Bleu para se manterem secos, se começar a chover.

A Winona e a Wickerbottom já estão terminando um sistema de pedidos para rodar no Winona9000 (que faz o pré-processamento da entrada), mas falta o módulo que decide quais receitas devem ser feitas na ordem correta. Cada sobrevivente pode fazer um pedido por vez, além de alterações de prioridade no pedido que ainda não foi entregue. Além disso, o lugar que eles estão implica em diferentes configurações de cozinhas, com quantidades diferentes de



panelas, o que impacta na quantidade de pedidos que podem ser feitos em paralelo. Ou seja, a quantidade de panelas indica a quantidade de pedidos entregues por “rodada”.



## Entrada

A primeira linha da entrada contém os valores  $1 \leq P, S \leq 10^5$  que representam a quantidade de panelas e a quantidade de sobreviventes, respectivamente. As  $S$  linhas seguintes contêm os nomes e sobrenomes (separados por um espaço, com no máximo 15 caracteres cada) dos sobreviventes que podem ou não fazer pedidos e um número inteiro  $0 < Q \leq 650000$  que indica a quantidade de dias sobrevividos. Assim, o formato destas linhas é:

NOME SOBRENOME QUANTIDADE

As linhas seguintes são compostas de um inteiro  $N$ , indicando a quantidade de pedidos e/ou alterações feitos nessa rodada, descritos nas  $N$  linhas seguintes. Cada operação usa o ID do sobrevivente, que é um valor de  $0$  à  $S-1$  na mesma ordem que os nomes foram dados na entrada. As opções são dadas no formato:

novo <ID> <PRATO>

para indicar que o sobrevivente com ID <ID> fez um novo pedido de <PRATO>. O nome do prato é uma *string* que pode ter mais de uma palavra separada com espaços, com no máximo 25 caracteres; Outra opção é:

altera <ID> <VALOR>

para indicar que o pedido feito pelo sobrevivente <ID> recebeu uma alteração de prioridade. O <VALOR> (que pode ser positivo ou negativo) é somado ao valor atual. Nesse caso, a prioridade do pedido atual muda, mas o próximo pedido vai ter o valor original da entrada para esse sobrevivente. A entrada acaba quando N = 0.

## Saída

Cada bloco de N pedidos e/ou alterações (da segunda parte da entrada) representa uma rodada. No final de cada rodada, os P pedidos com maior prioridade são preparados e entregues pelo chef. A saída de cada rodada começa com

---- rodada %d ----

Seguido de P linhas, contendo cada pedido entregue no formato:

<NOME> <SOBRENOME> <PRATO> <PRIORIDADE ATUAL>

Se, no fim de uma rodada, a quantidade de pedidos for menor do que a quantidade de panelas disponíveis, os pedidos são entregues normalmente (ou seja, esta rodada terá menos que P linhas na saída). Lembre-se que a ordem da saída é do maior para o menor. Se quando N = 0 ainda houverem pedidos não entregues, as rodadas de entrega continuam acontecendo, mesmo sem pedidos novos.

Se houverem dois pedidos com prioridades iguais, o pedido do sobrevivente com menor ID é entregue primeiro.

## Exemplos

Exemplo 1:

**Entrada**

```
2 1  
Wolfgang Strongman 1080  
1  
novo 0 Creamy Potato Purée  
1  
novo 0 Fancy Spiralled Tubers  
1  
novo 0 Puffed Potato Soufflé  
0
```

### Saída

```
---- rodada 1 ----  
Wolfgang Strongman Creamy Potato Purée 1080  
---- rodada 2 ----  
Wolfgang Strongman Fancy Spiralled Tubers 1080  
---- rodada 3 ----  
Wolfgang Strongman Puffed Potato Soufflé 1080
```

### Exemplo 2:

#### Entrada

```
3 3  
Winona Handywoman 110  
Wickerbottom Librarian 120  
Wes Silent 649900  
3
```

```
novo 1 Surf 'n' Turf  
novo 0 Spicy Vegetable Stinger  
novo 2 Fresh Fruit Crepes  
0
```

Saída

```
---- rodada 1 ----  
Wes Silent Fresh Fruit Crepes 649900  
Wickerbottom Librarian Surf 'n' Turf 120  
Winona Handywoman Spicy Vegetable Stinger 110
```

### Exemplo 3:

#### Entrada

```
1 4  
Woodie Lumberjack 800  
Maxwell Puppet 220  
Wilson Scientist 450  
Wendy Bereaved 102470  
3  
novo 3 Banana Pop  
novo 1 Wobster Dinner  
altera 1 200000  
3
```

```
novo 1 Meatballs  
altera 3 -102300  
novo 2 Bacon and Eggs  
0
```

## Saída

```
---- rodada 1 ----  
Maxwell Puppet Wobster Dinner 200220  
---- rodada 2 ----  
Wilson Scientist Bacon and Eggs 450  
---- rodada 3 ----  
Maxwell Puppet Meatballs 220  
---- rodada 4 ----  
Wendy Bereaved Banana Pop 170
```

## Regras e Avaliação

Para este lab, o uso de funções de ordenação (insertionsort, mergesort, selectionsrt, etc.) não é permitido.

Seu código será avaliado não apenas pelos testes do GitHub, mas também pela qualidade. Dentre os critérios subjetivos de qualidade de código analisaremos neste laboratório:

- A escolha de bons nomes de funções e variáveis;
- A ausência de trechos de código repetidos desnecessariamente;
- O uso apropriado de funções;
- A ausência de vazamentos de memória;
- A eficiência dos algoritmos propostos;
- A correta utilização das Estruturas de Dados;
- Você deve implementar uma estrutura de heap binária para gerenciar a prioridade dos pedidos

Note, porém, que essa não é uma lista exaustiva, pois outros critérios podem ser analisados dependendo do código apresentado visando mostrar ao aluno como o código poderia ser melhor.

## Submissão

Você deverá submeter sua solução no repositório criado no aceite da tarefa. Você pode enviar arquivos adicionais caso deseje para serem incluídos por `cozinha.c`.

Não se esqueça de dar `git push` !

**Atenção:** O repositório da sua atividade conterá alguns arquivos iniciais. Fica **estritamente proibido** ao aluno alterar os arquivos já existentes, tais como o testador existente ou demais arquivos de configuração do laboratório.

Lembre-se que sua atividade será corrigida automaticamente na aba “Actions” do repositório. Confirme a correção e o resultado, já que o que vale é o que está lá e não na sua máquina.

Após a correção da primeira entrega, será aberta uma segunda chance, com prazo de entrega apropriado.