MC202AB - Estrutura de Dados

Lab 09 - Heap

Data da Primeira Chance: 31 de outubro de 2022
Peso: 4

Jennifer Walters (agora conhecida como *She-Hulk*) é uma advogada em ascensão de carreira que acabou de obter o cargo principal de defensora em casos criminais que envolvam pessoas com superpoderes.

Ao começar nesse cargo, ela se deparou com uma certa bagunça dos processos empilhados na mesa dela, então ela decidiu organizar esses processos digitalizando eles e armazenando, principalmente, o número identificador P (500 ≤ P ≤ 10000) do processo e o código C (1 ≤ C ≤ 500) que representa a prioridade deste processo de acordo com o chefe de Jennifer, ou seja, quanto menor o valor de C, maior a prioridade do processo P que ele representa. Pode haver processos com o mesmo valor C de prioridade, então para esses casos o identificador P é usado para desempate (quanto maior o valor de P, maior a prioridade do processo).

Após ter digitalizado todos esses processos e separá-los em grupos criminais, agora ela precisa obter os números dos processos mais prioritários para começar a analisá-los. Porém, ela precisa de ajuda com essa busca, pois a quantidade de processos é absurdamente grande, e existe a dificuldade de que a urgência destes processos é dinâmica, isto é, o código C que representa a prioridade do processo P pode mudar para mais ou para menos no decorrer do tempo.

Sua tarefa neste lab é ajudar Jennifer Walters a buscar os K (2 ≤ K ≤ 100) processos mais prioritários em um conjunto de processos através de buscas em determinada lista de pares (P, C) utilizando o conceito de ***fila de prioridade***.

Entrada

Na entrada do programa serão dados o tamanho T (5 ≤ T ≤ 500) da lista de pares (P, C), seguido da lista em si, em que cada par será dado em uma única linha no formato "$[P]\ [C]$" (número P separado por um espaço " " do número C), por exemplo:

```
5
2541 23
543 2
9271 6
17623 1
1565 3
```

Em seguida, será dado um valor inteiro Q (5 ≤ Q ≤ 100) em uma única linha, representando a quantidade de comandos que serão solicitados em seguida.

Depois disso, serão dados os comandos para remover da fila de prioridade os K processos mais prioritários na forma "$R\ [K]$" (comando separado por um espaço " " do número K), por exemplo:

```
R 2
```

Também serão dados comandos para modificar a prioridade de determinado processo de forma intercalada com o comando acima, no formato "$M\ [P]\ [C]$" (comando separado por um espaço " " do número P do processo, separado por um espaço " " do novo valor C de prioridade do processo P), por exemplo:

```
M 543 6
```

No exemplo acima, supondo que o processo 543 tinha a prioridade 2, agora ele tem a prioridade 6, fazendo com que a fila de prioridades possa ser modificada por consequência disso também.

Saída

A saída do programa deverá ser os números dos processos removidos da fila de prioridade a cada vez que o comando R for executado, seguindo o formato "$PROCESSOS\ REMOVIDOS:\ [P_1]\ [P_2]\ ...\ [P_K]$" em que K é a quantidade de processos removidos da fila de prioridade. Após finalizar o processo de modificações e remoções, a mensagem "$FINALIZADO!$" deverá ser exibida. Por exemplo:

```
PROCESSOS REMOVIDOS: 1575 2674 2149 245
```

```
PROCESSOS REMOVIDOS: 23545 2124
PROCESSOS REMOVIDOS: 23951 315856 841235
FINALIZADO!
```

Exemplos

Exemplo 1:

Entrada

```
5
9828 10
5808 5
4972 2
3217 12
7293 3
3
R 2
M 3217 6
R 2
```

Saída

```
PROCESSOS REMOVIDOS: 4972 7293
PROCESSOS REMOVIDOS: 5808 3217
FINALIZADO!
```

Exemplo 2:

Entrada

```
10
1515 12
3792 5
4854 30
893 21
9857 11
1410 2
3445 102
4666 206
7199 1
5810 26
8
M 1515 4
R 3
M 9857 58
M 4666 57
M 3445 1
R 3
```

```
M 4854 63
R 2
```

Saída

```
PROCESSOS REMOVIDOS: 7199 1410 1515
PROCESSOS REMOVIDOS: 3445 3792 893
PROCESSOS REMOVIDOS: 5810 4666
FINALIZADO!
```

Regras e Avaliação

Importante: é obrigatório o uso dos conceitos de ***HEAP*** para implementar a solução deste lab.

Seu código será avaliado não apenas pelos testes do CodePost, mas também pela qualidade. Dentre os critérios subjetivos de qualidade de código iremos analisar: o uso apropriado de funções, de comentários, e de structs; a escolha de bons nomes de funções, variáveis e de structs e seus campos; o correto uso de Tipos Abstratos de Dados e a correta separação em vários arquivos; a ausência de diversos trechos de código repetidos, e o tempo de execução e uso de memória dos algoritmos projetados. Note, porém, que essa não é uma lista exaustiva, pois outros critérios podem ser analisados dependo do código apresentado visando mostrar ao aluno como o código poderia ser melhor.

Submissão

Você deverá submeter **três arquivos** na tarefa *Lab 09 - Heap* no CodePost: ***heap.h***, ***heap.c, lab09.c***. No arquivo ***heap.h*** é esperado que você declare a *struct* com os devidos campos e funções necessárias para implementar os comportamentos da fila de prioridade. No arquivo ***heap.c*** é esperado que você implemente as funções declaradas no arquivo ***heap.h*****.** Por fim, no arquivo ***lab09.c*** é esperado que o faça a leitura das entradas do programa e execute as chamadas de funções por meio do TAD implementado.

Após o prazo estabelecido para a atividade, será aberta uma tarefa Lab 09 - Segunda Chance, com prazo de entrega até o fim do semestre.