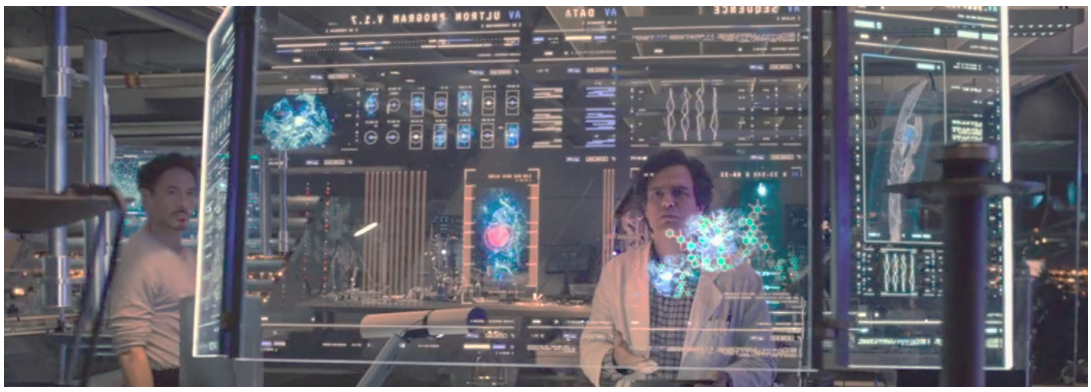
MC202AB - Estrutura de Dados

Lab 05 - Vetores dinâmicos

Data da Primeira Chance: 26 de setembro de 2022
Peso: 3

Tony Stark e Bruce Banner ainda continuam na tarefa de criar o sistema Ultron. Agora estão em uma fase em que Dr. Banner conseguiu calcular valores neurais para que a consciência única do sistema funcione. Porém, Tony ainda precisa testar certas combinações matriciais destes valores.



A tarefa de Tony Stark é fazer um TAD (Tipo abstrato de dados)[1] para trabalhar com uma matriz de inteiros, porém ela precisa mudar de tamanho dinamicamente, pois serão adicionados e removidos valores dela conforme o teste de combinação matricial que ele planeja fazer.

Dr. Banner fornecerá uma matriz com valores ordenados nas linhas e após isso, dados os parâmetros de posição, serão solicitadas as seguintes operações (não necessariamente nesta ordem, e com repetição de operações):

- **`IN L [valores separados por " "]`** - Comando para inserção de linha na matriz, em que a linha deve inserida na última posição da matriz;
    - Obs.: Se no momento desta operação a matriz tiver *N* colunas, a quantidade de valores da nova linha inserida também será *N.*

- **`OUT L [X]`** - Comando para remoção da linha na posição *X* da matriz, fazendo com que todas as linhas alocadas em posições após o *X* passam a ter índice subtraído de 1 por causa da remoção;

- **`IN C [valores separados por " "]`** - Comando para inserção de coluna na matriz, em que cada valor deve ser inserido em cada linha da matriz com a <u>atenção para manter a ordem dos valores</u> já existentes em cada linha;

[1] Verificar seção “Submissão” para mais detalhes dos arquivos deste TAD.

    - Obs.: Se no momento desta operação a matriz tiver *M* linhas, a quantidade de valores da nova coluna inserida também será *M.*

- **`OUT C [Y]`** - Comando para remoção de coluna, em que todos os valores neurais na posição Y de cada linha deve ser removido, fazendo com que todos os valores em posições após o Y passam a ter índice subtraído de 1 por causa da remoção;

Para otimizar seu código, Tony Stark definiu que o espaço alocado de uma linha ou de uma coluna deve **dobrar** quando a linha ou coluna estiver cheia. Assim como, o espaço alocado deve cair pela **metade** quando somente 1/4 das posições alocadas estiverem cheias.

Mais explicitamente, esta regra define que a quantidade de memória alocada para o tamanho da linha da matriz vai aumentar e diminuir, e a quantidade de memória alocada para o número de vetores nesta matriz (isto é, número de linhas) vai aumentar e diminuir também.

A outra regra definida é que cada linha da matriz se mantenha ordenada após cada operação realizada, então é preciso ter bastante atenção ao inserir valores nesta matriz.

Como aluno de MC202 você deve fazer o papel de Tony Stark e implementar esta tarefa para continuar a construção do sistema Ultron.

Dicas para evitar vazamento de memória:

- Toda vez que a função *malloc* for utilizada, o *free* também precisa ser chamado;
- Não deixe ponteiros “orfãos”: Se você utilizar um ponteiro para alocar memória para uma *struct*, depois aloca memória para uma variável interna dessa *struct*, primeiro você faz *free* para o ponteiro de dentro da *struct* e depois o *free* para o ponteiro da *struct* em si (“from child to parent”);
- Escreva comentários acima das funções, isso pode te ajudar a lembrar onde alocou e desalocou memória;
- Faça o chamado “Teste de mesa” com seu código, pegando um teste bem pequeno e anotar no seu caderno passo-a-passo do que está acontecendo no seu código;
- **Evite fazer todo o código de uma única vez**: Faça um trecho, teste. Faça outro trecho, teste de novo, e assim por diante. Isso melhora seu tempo na hora de procurar saber onde está o erro.

Entrada

Primeiramente será dado o tamanho inicial da matriz informando a quantidade de linhas e a quantidade de colunas em uma única linha separados por “ “, e em seguida a matriz com valores de cada linha separados por um único “ “, como mostra o exemplo abaixo:

```
3 4
50 62 85 87
20 26 34 35
12 13 28 29
```

Em seguida, será passada a quantidade de operações que dizem respeito a combinação matricial que Tony Stark deseja fazer, seguido das operações e seus respectivos parâmetros, como mostra o exemplo abaixo:

```
2
IN C 3 57 6
IN L 23 25 28 29
```

Atenção: A matriz inicial é quem vai ser alterada sempre, então cada operação depende do resultado da operação anterior.

Saída

A saída desta tarefa é imprimir na tela a matriz resultante de cada operação realizada, seguido da separação "`###`" em uma única linha. Após todas as operações, imprimir na tela a seguinte frase:

```
COMBINACAO MATRICIAL FINALIZADA!
```

Exemplos

Exemplo 1:

Entrada

```
4 3
12 51 64
40 53 62
26 45 46
10 11 13
1
IN C 54 22 30 21
```

Saída

```
12 51 54 64
22 40 53 62
26 30 45 46
10 11 13 21
###
COMBINACAO MATRICIAL FINALIZADA!
```

Exemplo 2:

Entrada

```
3 5
102 106 107 121 125
5 6 11 32 35
```

```
76 82 84 85 88
5
IN L 14 15 19 22 23
OUT C 1
OUT C 3
OUT L 0
IN L 3 1 5
```

Saída

```
102 106 107 121 125
5 6 11 32 35
76 82 84 85 88
14 15 19 22 23
###
102 107 121 125
5 11 32 35
76 84 85 88
14 19 22 23
###
102 107 121
5 11 32
76 84 85
14 19 22
###
5 11 32
76 84 85
14 19 22
###
5 11 32
76 84 85
14 19 22
3 1 5
###
COMBINACAO MATRICIAL FINALIZADA!
```

Regras e Avaliação

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 (**obrigatório neste lab!**) 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 dependendo do código apresentado visando mostrar ao aluno como o código poderia ser melhor.

Submissão

Você deverá submeter **<u>três arquivos</u>** na tarefa *Lab 5 - Vetores dinâmicos* no CodePost: ***ultron.h***, ***ultron.c*** e ***lab05.c***. No arquivo ***ultron.h*** é esperado que você declare a *struct* com os devidos campos necessários para uma alocação dinâmica vista em aula, assim como você precisa declarar as funções de inserção e remoção, de criação da matriz, e outras funções que achar necessário. No arquivo ***ultron.c*** é esperado que você implemente as funções declaradas no ***ultron.h*.** Por fim, no arquivo ***lab05.c*** é esperado que o faça a leitura das entradas do programa execute as chamadas de funções por meio do TAD implementado.

ATENÇÃO: A ferramenta *valgrind* será usada para analisar vazamentos de memória na execução do seu código e, com isso, alguns <u>pontos podem ser decrementados</u> do seu lab dependendo do resultado do *valgrind.*

Após o prazo estabelecido para a atividade, será aberta uma tarefa Lab 5 - Segunda Chance, com prazo de entrega até o fim do semestre.