IMD0029 - Estruturas de Dados 1 - 2023.2 - Unidade 2

Prof. Eiji Adachi

LEIA ANTES DE COMEÇAR

- Esta é uma atividade de caráter individual.
- O valor de cada questão é informado no seu enunciado.
- Desvios éticos ou de honestidade levarão à nota igual a zero na respectiva unidade.
- É terminantemente proibido o acesso a qualquer tipo de material impresso ou eletrônico. Qualquer acesso (ex.: sites na Internet) resultará na anulação da prova e nota igual a zero na respectiva unidade.
- Celulares e outros dispositivos eletrônicos devem permanecer desligados e à vista durante toda a prova. Qualque consulta a dispositivos eletrônicos resultará na anulação da prova e nota igual a zero na respectiva unidade.
- Junto a este enunciado, você também recebeu uma estrutura de diretórios contendo um diretório para cada questão. Em cada um destes diretórios, já existe uma assinatura de função, um arquivo `Makefile` e um teste executável, que pode ser executado usando o comando `make run-test`.
- A solução da sua questão deverá seguir a assinatura da função já estabelecida, conforme o enunciado. Ou seja, não mude esta assinatura. Se necessário, crie funções auxiliares com outras assinaturas, mas não mude a assinatura da função original.

Critérios de correção

Para a correção desta atividade, serão levados em consideração, dentre outros, os seguintes pontos:

- Obediência às regras definidas para as assinaturas de função e para o entregável (arquivo .zip e estrutura de diretórios), conforme especificado no enunciado.
- Existência de erros ou *warnings* de compilação do código fonte. Compile usando o arquivo `Makefile` disponibilizado.
- Programas executam sem apresentar falhas e produzem os resultados esperados.
- Soluções atendem critérios de complexidade, caso estabelecido no enunciado.
- Apresentação e organização do código fonte entregue (identação, nome das variáveis, modularização do código em funções, etc).

Obs.: Para cada questão, já há um pequeno teste executável. Este é um teste simples que não garante a corretude da sua implementação. Ou seja, se sua implementação passou no teste executável disponibilizado junto a este enunciado, isto não é garantia de que ela está totalmente correta. Para fins de correção, eu utilizarei outra bateria de testes mais completa, além de analisar manualmente o código produzido.

Entregável

O entregável desta atividade deverá seguir a mesma estrutura de diretórios do código fonte que você recebeu com este enunciado, obviamente, contendo os arquivos fonte utilizados para construir sua solução nos diretórios de cada questão. Além disso, o diretório raiz deverá ter o seu nome, seguindo o padrão `<PRIMEIRO_NOME>_<SOBRENOME>`

Por exemplo:

```
> JOAO_SILVA
>   lib
>   q1
>   q2
>   q3
```

Toda esta estrutura de diretórios, incluindo os arquivos fonte com sua solução, deverá ser compactada num arquivo .zip (não é .rar, nem .tar.gz, nem qualquer outra extensão) que também deverá seguir o padrão `<PRIMEIRO_NOME>_<SOBRENOME>.zip` .

Caso você não tenha resolvido alguma questão, não coloque o seu respectivo diretório no entregável. Ou seja, remova o diretório da questão que você não fez antes de compactar o arquivo entregável.

O .zip não deve conter qualquer arquivo executável ou código objeto (arquivo com extensão `.o`). Execute o comando `make clean` antes de compactar o seu entregável para remover esses arquivos.

O arquivo compactado deverá ser entregue via SIGAA até o horário estabelecido na tarefa. Este é um prazo fixo que não será estendido, exceto em casos muito excepcionais (ex.: SIGAA fora do ar). Ou seja, entregas após este horário não serão aceitas. Certifique-se de que está enviando a versão correta antes de anexar ao SIGAA.

Questão 1 - Valor: 2.0

Dada uma lista simplesmente encadeada com ponteiro para o início, implemente o seguinte método que rotaciona à direita os elementos da lista:

```
void  ListaEncadeada::rotacionar(int x)
```

Esse método implementa a rotação dos elementos de uma lista à direita. Rotacionar uma lista 1 vez à direita significa mover o último elemento da lista para o início da lista. Similarmente, rotacionar uma lista `x` vezes à direita significa mover os últimos `x` elementos da lista para o início da lista. Por exemplo: considerando a lista `l = {1 -> 3 -> 7 -> 4 -> null}`:

- se `x = 1` (1 rotação à direita) então o resultado esperado para a operação rotacionar é `l = {4 -> 1 -> 3 -> 7 -> null}`;
- se `x = 2` (2 rotações à direita) então o resultado esperado para a operação rotacionar é `l = {7 -> 4 -> 1 -> 3 -> null}`.

Obs.: Essa questão deve ser resolvida apenas manipulando os nós da lista, i.e., desencadeando e encadeando os nós nos locais adequados; não é permitido alocar memória para novos nós, tampouco liberar memória de nós já criados previamente.

Questão 2 - Valor: 1.5

Dada uma lista simplesmente encadeada com ponteiro para o início, implemente o seguinte método que remove elementos repetidos da lista:

```
int  ListaEncadeada::removerRepetidos()
```

Esse método remove os elementos da lista que possuem repetição, retornando o número de elementos que foram removidos da lista. Considere que a lista já está ordenada quando o método `removerRepetidos` é invocado. Considere também que se um elemento repetido estava na lista, apenas uma instância sua deverá permanecer na lista após a execução do método `removerRepetidos`. Por exemplo: considere a lista `l = {1 -> 1 -> 3 -> 7 -> 7 -> 7 -> 14 -> null}`. Após a execução do método `removerRepetidos`, a lista estará no seguinte estado: `l = {1 -> 3 -> 7-> 14 -> null}` e o método retornar 3, pois foram removidos um `1` e dois `7`. Sua solução deverá ter complexidade de tempo da ordem de O(n).

Obs.: Não é permitido usar estruturas auxiliares, como stack, array ou vector, por exemplo.

Questão 3 - Valor: 1.5

Dada uma lista simplesmente encadeada com ponteiro para o início, implemente o seguinte método que inverte uma lista:

```
void ListaEncadeada::inverter()
```

O método `inverter` inverte a ordem dos elementos da lista. Por exemplo: considere a lista `l = {1 -> 7 -> 3 -> 9 -> null}`. Após a execução do método `inverter`, o estado da lista deverá ser `l = {9 -> 3 -> 7 -> 1 -> null}`.

Obs.: Não é permitido usar estruturas auxiliares, como stack, array ou vector, por exemplo. Também não é permitido deletar ou criar novos nós; deve-se apenas manipular os nós e seus ponteiros que já existem.