IMD0029 - Estruturas de Dados 1 - 2023.2 - Unidade 2

Prof. Eiji Adachi

LEIA ANTES DE COMEÇAR

- Esta é uma atividade de caráter individual e sem consultas a pessoas ou material (impresso ou eletrônico).
- O valor de cada questão é informado no seu enunciado.
- Celulares e outros dispositivos eletrônicos devem permanecer desligados durante toda a prova.
- Desvios éticos ou de honestidade levarão à 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: 1.5

Dada uma lista duplamente encadeada com sentinelas cabeça e cauda, implemente o seguinte método:

```
bool ListaDuplamenteEncadeada::inserirOrdenado(std::string s)
```

Esse método insere um elemento na lista, mantendo-a ordenada em ordem **DECRESCENTE**. Considere que a lista ou está vazia ou já está ordenada. Caso o elemento já exista na lista, a função não cria um novo nó e retorna `false`; caso o elemento ainda não exista na lista, deve-se criar um novo nó e encadeá-lo na lista, mantendo-a ordenada, incrementando a quantidade de elementos da lista e retornando `true`.

Para esta questão, implemente o método `inserirOrdenado` já declarado no arquivo `src/ListaDuplamenteEncadeada.cpp`. Sua solução deverá ter complexidade de tempo `O(n)`.

Questão 2 - Valor: 2.0

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

```
bool  ListaEncadeada::remover(std::string val)
```

Esse método implementa a remoção do nó cujo valor é igual ao parâmetro de entrada `val`. O método deve retornar `true` caso algum nó seja removido, e `false` caso contrário (i.e., não existe nó na lista com valor igual a `val`). Lembre-se de decrementar a quantidade de nós da lista e de liberar a memória do nó removido.

Questão 3 - Valor: 1.5

Dada uma lista simplesmente encadeada com ponteiro para o início da lista (primeiro nó), implemente o seguinte método que verifica se uma lista é simétrica ou não:

```
bool ListaEncadeada::e_simetrica()
```

Uma lista é dita simétrica se ela apresenta a mesma sequência de elementos quando lida no sentido da esquerda para direita quanto no sentido da direita para a esquerda. Por exemplo, a lista encadeada: "alpha" <-> "bravo" <-> "charlie" <-> "bravo" <-> "alpha" é simétrica porque possui a mesma sequência de elementos quando lida em ambos os sentidos, enquanto a lista "alpha" <-> "bravo" <-> "charlie" <-> "delta" <-> "alpha" não é. Considere que listas vazias e listas com um só elemento são simétricas. A sua implementação deverá ter complexidade de tempo `O(n)` e deverá, **obrigatoriamente, usar uma Pilha**.

Para usar uma Pilha, você deve usar a estrutura `stack` da biblioteca padrão de C++:

- Use o método push para inserir um elemento na pilha (ex.: `pilha.push("A");`)
- Use o método pop para remover um elemento da pilha (ex.: `pilha.pop();`)

- Use o método top para acessar o elemento no topo da pilha (ex.: `std::string topo = pilha.top();`)
- Use o método empty para verificar se a pilha está vazia (ex.: `if( pilha.empty() )`)
- Use o método size para saber o tamanho (quantidade de elementos) da pilha (ex.: `size_type tamanho = pilha.size();`)

Veja o arquivo `src/main.cpp` para ver um exemplo de como usar uma `stack`.