IMD0029 - EDB1 - 2024.2 - Unidade 2

Prof. Eiji Adachi

LEIA ANTES DE COMEÇAR

- Atividade individual sem consulta a pessoas ou materiais (impresso ou eletrônico).
- O valor de cada questão está indicado no enunciado.
- Mantenha celulares e outros eletrônicos desligados durante a prova.
- Desvios éticos resultarão em nota zero nesta unidade.
- Você recebeu diretórios para cada questão, cada um contendo uma assinatura de função, um arquivo `Makefile` e um teste executável, que pode ser executado usando o comando `make run-test`. O teste executável não garante a total corretude da sua implementação.
- Siga a assinatura de função fornecida; não a altere. Se necessário, crie funções auxiliares, mas mantenha a assinatura original.

Critérios de Correção

Na correção desta atividade, serão considerados:

- **Obediência às regras** das assinaturas de função e do entregável (arquivo .zip e estrutura de diretórios), conforme especificado.
- **Ausência de erros ou warnings** na compilação do código-fonte. Use o `makefile` para compilar.
- **Execução correta dos programas**, sem falhas e produzindo os resultados esperados.
- **Atendimento aos critérios de complexidade**, se estabelecidos no enunciado.
- **Qualidade do código-fonte**: boa apresentação, organização, identação, nomes de variáveis adequados, modularização em funções, etc.

Entregável

- O entregável deve manter a mesma estrutura de diretórios do código-fonte recebido, contendo os arquivos de solução em cada diretório de questão.
- O diretório raiz deve ter seu nome, no formato `PRIMEIRO_NOME_SOBRENOME`. Exemplo:

```
JOAO_SILVA
├── q1
├── q2
└── q3
```

- Compacte toda a estrutura em um arquivo `.zip` (não use `.rar`, `.tar.gz` ou outros), seguindo o mesmo padrão de nome: `PRIMEIRO_NOME_SOBRENOME.zip`.
- O `.zip` não deve conter arquivos objeto e executáveis; remova-os antes de compactar.
- Entregue o arquivo via SIGAA até o horário estabelecido. Entregas após o prazo só serão aceitas em casos excepcionais (ex.: SIGAA fora do ar).

Questão 1 - Valor: 1.5

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

```
bool ListaDuplamenteEncadeada::inserirAntes( std::string novoElemento,
std::string referencia )
```

Esse método recebe duas strings como parâmetros de entrada e realiza a seguinte operação:

- se algum nó na lista tiver conteúdo igual a `referencia`, então um novo nó com conteúdo `novoElemento` é criado e encadeado imediatamente antes do **primeiro** nó com conteúdo `referencia`, incrementando a quantiadade de elementos da lista e retornando `true`;
- se nenhum nó da lista tiver conteúdo igual a `referencia`, então a função não insere um novo nó na lista e retorna `false`.

Exemplo de Uso: Suponha que a lista atual contenha os elementos: `"D" <-> "B" <-> "A"`.

- Chamada `inserirAntes("C", "B")` resultará na lista: `"D" <-> "C" <-> "B" <-> "A"` e retornará `true`.
- Chamada `inserirAntes("F", "E")` retornará `false` pois `"E"` não está na lista.

Para esta questão, implemente o método `inserirAntes` já declarado no arquivo `src/ListaDuplamenteEncadeada.cpp`.

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

Sua solução deverá ter complexidade de tempo θ(n).

Questão 2 - Valor: 2.0

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

```
int ListaEncadeada::removerTodos(int val)
```

Esse método remove todos os elementos da lista que possuem valor igual a `val`, retornando o número de elementos que foram removidos da lista.

Por exemplo: considere uma lista no seguinte estado `{1 -> 7 -> 3 -> 7 -> 5 -> 7 -> 14 -> null}`. Após a execução da chamada `removerTodos(7)`, a lista estará no estado `{1 -> 3 -> 5 -> 14 -> null}` e o método retornará 3, pois foram removidos três nós cujos valores eram `7`.

Sua solução deverá ter complexidade de tempo θ(n).

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

Questão 3 - Valor: 1.5

Nesta questão, implemente a seguinte função:

```
bool ehPalindromo(string str);
```

Este método recebe como entrada uma string e verifica se esta string é palíndromo ou não. Uma string é um palíndromo se ela lê da mesma forma de frente para trás e de trás para frente, ignorando espaços e pontuações. Por exemplo:

- "ana" é palíndromo.
- "arara" é palíndromo.
- "a man a plan a canal panama" é palíndromo (ignorando espaços).

Sua solução deve ignorar espaços em branco na verificação se é ou não palíndromo. Considere que não serão usadas como entrada para a função strings com pontuações nem com letras acentuadas.

Neste questão, sua solução deverá, obrigatoriamente, usar uma pilha. Para isso, use uma `stack` da biblioteca padrão de C++. No arquivo `palindromo.cpp` há um exemplo de como iterar sobre os caracteres da string de entrada, como identificar um espaço em branco e como usar uma pilha do tipo `stack`.