IMD0029 - Estrutura de Dados Básicas 1 – 2018.2
Prof. Eiji Adachi M. Barbosa
Atividade Avaliativa Prática em Laboratório – Unidade 2

ANTES DE COMEÇAR, leia atentamente as seguintes instruções:

- Esta é uma atividade de caráter individual e sem consultas a pessoas ou material (impresso ou eletrônico).
- A atividade vale 5,0 pontos na 1ª unidade. 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 anulação da atividade do candidato (nota igual a zero).
- 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 e uma função main com um pequeno teste executável. A solução da sua questão deverá seguir a assinatura da função já estabelecida. Ou seja, não mude esta assinatura. Se necessário, crie funções auxiliares com outras assinaturas, mas **não mude a assinatura da função original!**

Questão 1 (1,5 ponto): Considere a classe `Queue` do diretório /q1, a qual define a estrutura básica para uma **Fila**. Nesta questão, implemente os métodos `Queue<T>::peek()`, `Queue<T>::enqueue(T element)` e `Queue<T>::dequeue()` reusando a pilha (stack) que é provida pela Standard Template Library (STL) de C++.

O código a seguir exemplifica o uso de uma stack da STL:[1]

```
1   // stack::push/pop
2   #include <iostream>      // std::cout
3   #include <stack>         // std::stack
4
5   int main ()
6   {
7     std::stack<int> mystack;
8
9     for (int i=0; i<5; ++i) mystack.push(i);
10
11    std::cout << "Popping out elements...";
12    while (!mystack.empty())
13    {
14      std::cout << ' ' << mystack.top();
15      mystack.pop();
16    }
17    std::cout << '\n';
18
19    return 0;
20  }
```

Questão 2 (2,0 pontos): Nesta questão, a classe `LinkedList` do diretório /q2 define a estrutura básica para uma lista duplamente encadeada com sentinelas cabeça (`Head`) e cauda (`Tail`). Implemente o método `LinkedList::sort()`, o qual realiza a ordenação da lista. O método de ordenação a ser implementado deverá ser o *Bubble Sort*.

Obs.: Antes de implementar o método `sort`, é necessário implementar o método `insertEnd`.

[1] Código extraído da página: http://www.cplusplus.com/reference/stack/stack/push/

Questão 3 (1,5 ponto): Uma sequência de parênteses, colchetes e chaves é bem formada quando o abre-fecha de parênteses, colchetes e chaves é feito na ordem e quantidade corretas. Por exemplo, a sequência de parênteses e colchetes a seguir é bem formada `[ ( ) ( ( ) ) ]` enquanto a sequência `[ ( ] )` não é bem formada. A primeira sequência é bem formada pois o abre-fecha de parênteses, colchetes e chaves está correto, enquanto na segunda sequência o colchete foi fechado antes de se fechar o parênteses. Nesta questão, implemente a função `isWellFormed(string str)`, a qual recebe uma string representando uma sequência de parênteses, colchetes e chaves e verifica se esta sequência é bem formada ou não. Use uma Pilha (`Stack`) da STL para resolver esta questão.
Obs.: Pode assumir nesta questão que os caracteres da string de entrada serão apenas parênteses, colchetes ou chaves. Assuma também que parênteses, colchetes e chaves podem aparecer em qualquer ordem, isto é, não existe uma ordem pré-estabelecida de prioridade entre estes elementos em sequências bem formadas.

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 pai deverá ter o seu nome e matrícula, seguindo o padrão `<PRIMEIRO_NOME>_<SOBRENOME>-<MATRICULA>` (Dica: basta renomear o diretório /src com o padrão definido anteriormente). Por exemplo:

```
> JOAO_SILVA-200012345
   > q1
   > q2
   > q3
   > q4
```

Toda esta estrutura de diretórios, incluindo os arquivos fonte com sua solução, deverá ser compactada num arquivo .zip que também deverá seguir o padrão `<PRIMEIRO_NOME>_<SOBRENOME>-<MATRICULA>.` Este arquivo compactado deverá ser entregue via SIGAA até as **20:25**. **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. A atividade do SIGAA permite apenas um envio, portanto certifique-se de que está enviando a versão correta antes de anexar ao SIGAA.

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), conforme especificado no enunciado desta atividade
- Existência de erros ou warnings de compilação do código fonte[2]
- 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 uma função main com 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.

[2] Compile usando as flags `-Wall -pedantic -std=c++11`