INSTITUTO FED. DE EDUCAÇÃO, CIÊNC. E TEC. DE PERNAMBUCO
CURSO: TEC. EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
DISCIPLINA: ALGORITMOS E ESTRUTURAS DE DADOS
PROFESSOR: RAMIDE DANTAS
ASSUNTO: PILHAS E FILAS.

Prática 03

Parte 1: Pilhas

Problema 1: Resolva o problema Validação de Parênteses (Valid Parentheses, LeetCode 20). Dada uma string contendo caracteres de abertura e fechamento de parênteses '(' & ')', colchetes '[' & ']' e chaves '{' & '}', diga se a string é válida. Exemplos:

"[{()}]" : válida "()[]{}" : válida

"{[(}])" : inválida "(][){}" : inválida

O procedimento deve usar uma pilha (pode ser `std::stack<char>` ou `std::vector<char>`) para verificar a validade. Ao encontra um caractere de abertura '(', '[' ou '{', ele deve ter colocado na pilha. Ao encontrar um caractere de fechamento ')', ']' ou '}', o topo da pilha é desempilhado e deve corresponder ao caractere de fechamento, do contrário a string é inválida (se a pilha estiver vazia, também será inválida). Ao final, a pilha deve estar vazia para a string ser válida. Desconsidere a precedência entre parênteses, colchetes e chaves. Use o código abaixo para testar (submeta no LeetCode apenas o trecho destacado).

Desafio (Opcional): Resolva o problema de avaliação de expressões em Notação Polonesa Reversa (LeetCode 0150).

```
#include <iostream>
#include <stack>

using namespace std;

class Solution {
public:
    bool isValid(const string &s) {
       ...
    }
};

int main() {
    string testes[] = { "[{()}]", "()[]{}", "{[(}])", "(][){}", "}{)(][" };

    for (auto &s : testes) {
        cout << s << ": "<< (Solution().isValid(s)?"Valida":"Invalida")<<endl;
    }
}
```

Problema 2: Resolva o problema Contaminação (BC1583, resolvido na Prática 2) com uma **pilha** em vez de recursão, conforme explicado em sala. Use o código do problema Chuva (GitHub) como referência.

Parte 2: Filas

Problema 1: Implemente uma fila especial que retorna a média atual dos elementos presentes nela. Essa fila será usada no código abaixo para calcular a média móvel dos últimos N elementos (N = 4 no código). Implemente a fila de forma eficiente: use *buffer* circular, como nos slides, e com inteligência para não ter que percorrer toda a fila para recalcular a média.

```
#include <iostream>
#include <vector>

using namespace std;

class fila_media {
private:
    vector<int> itens;
    // sugestao de atributos; pode/deve haver outros
    int ini = 0,   // posicao do 1o elemento
        tam = 0,   // tamanho (numero atual de elementos)
        cap = 0;   // capacidade (numero max. de elementos)

public:
    // construtor receba capacidade; inicializa cap e o vector itens.
    explicit fila_media(int cap) : cap(cap), itens(cap) {  }

    bool cheia() const { ... } // fila esta cheia?

    bool vazia() const { ... } // fila esta vazia?

    void desenfileira() { ... } // remove elemento na frente da fila

    void enfileira(int i) { ... } // adiciona i ao final da fila

    int proximo() { ... } // retorna elemento na frente da fila

    double media() const { ... } // retorna atual media dos elementos

    int tamanho() { ... } // retorna tamanho atual da fila
};

int main() {
    int arr[] = {10, 2, 3, 5, 6, 10, 7, 9, 2, 6, 3, 13, 6};

    fila_media fila(4);

    for (int i : arr) {
        if (fila.cheia()) fila.desenfileira();

        fila.enfileira(i);

        cout << fila.media() << endl;
    }

    // Saida: 10 6 5 5 4 6 7 8 7 6 5 6 7

    return 0;
}
```

Problema 2: Resolva o problema Contaminação (BC1583, resolvido na Prática 2) com uma **fila** em vez de uma pilha ou recursão, conforme explicado em sala (isto é, algoritmo BFS ou busca em largura). Use o código do problema Chuva (GitHub) como referência.