CMP1054 - Estrutura de Dados I
4ª Lista de Exercícios - Pilhas
Max Gontijo de Oliveira

- Todos os métodos e funções criados nas questões deverão ser testadas em um programa principal (`main`).
- Caso haja necessidade, crie parâmetros adicionais para os métodos ou funções recursivos além dos explicitamente solicitados nas questões.
- Quando não estiver explícito na questão, o aluno poderá escolher entre criar um algoritmo iterativo ou recursivo.
- Em todas as questões, considere as classes `Node` e `Pilha` conforme visto em sala de aula.

```
class Node {
   public:
      TIPO x;
      Node *prox;
};

class Pilha {
        private:
                Node *topo;

        public:
                Pilha(); // Inicializa topo com NULL
                void push(TIPO); // empilhar
                TIPO pop();      // desempilhar
                TIPO get_top();  // obter elemento do topo da pilha
                bool is_empty(); // verifica se pilha está vazia
                void clear();    // limpar pilha
                int size();      // tamanho da pilha
                void inverter(); // inverte ordem dos elementos da pilha
                void print();    // imprimir elementos da pilha na tela

                // Outros métodos
};
```

- Na classe `Node` acima, `TIPO` se refere ao tipo de dado que a pilha deverá armazenar (`int`, `char`, `float`). Todavia, NÃO é obrigatório o uso de `template`. Assim, se o aluno preferir, pode criar uma classe `Node` e uma classe `Pilha` para cada questão.

1. Crie um MÉTODO chamado `push` na classe `Pilha` que receba um `int` e guarde esse valor no topo da pilha.

2. Crie um MÉTODO chamado `pop` na classe `Pilha` cujo retorno seja do tipo `int`; O método deverá retornar o valor do topo da pilha, removendo o respectivo `Node` da memória. Caso a pilha esteja vazia, o método deverá retornar 0.

3. Crie um MÉTODO chamado `get_top` na classe `Pilha` cujo retorno seja do tipo `int`; O método deverá retornar o valor do topo da pilha, sem remover o respectivo `Node` da memória. Caso a pilha esteja vazia, o método deverá retornar 0.

4. Crie um MÉTODO chamado `is_empty` na classe `Pilha` que retorne `true` se a pilha estiver vazia ou `false` caso contrário.

5. Crie um MÉTODO chamado `clear` na classe `Pilha` que remova todos os elementos da pilha.

6. Crie um MÉTODO chamado `size` na classe `Pilha` que retorne a quantidade de elementos da pilha.

7. Crie um MÉTODO chamado `inverter` na classe `Pilha` para alterar a ordem dos elementos da pilha. Dentro do método, instancie e utilize duas OUTRAS `Pilhas` para realizar essa mudança. Utilize apenas os métodos `push`, `pop` e `is_empty` da própria `Pilha` e das outras duas instâncias de pilha. NÃO é um método para imprimir os elementos. É um método para, efetivamente, inverter a ordem dos elementos na pilha.

8. Crie um MÉTODO chamado `print` na classe `Pilha` para imprimir todos os elementos da pilha.

9. Crie um MÉTODO na classe `Pilha` para imprimir todos os elementos da pilha **na ordem em que foram empilhados**. Utilize RECURSIVIDADE. NÃO utilize o método criado na questão 7.

10. Questão para testar sua pilha: crie um PROGRAMA que instancie uma classe `Pilha` (essa classe deve ter os métodos criados nas questões 1, 2, 4, 5, 6, 7, 8 e 9) e em seguida, execute os seguintes passos:

    1. *Programa lê do teclado um número inteiro OP.*

       (a) *Se $OP = 1$ o programa deverá ler um inteiro X e empilhar esse valor na pilha (`push`).*

   (b) *Se $OP = 2$ o programa deverá desempilhar o elemento do topo da pilha e exibir o valor na tela (`pop`).*
   (c) *Se $OP = 3$ o programa deverá exibir na tela o valor do elemento que está no topo da pilha (`pop`).*
   (d) *Se $OP = 4$ o programa deverá limpar a pilha (`clear`).*
   (e) *Se $OP = 5$ o programa deverá exibir a quantidade de elementos armazenados na pilha (`size`).*
   (f) *Se $OP = 6$ o programa deverá inverter a ordem dos elementos da pilha (`inverter`).*
   (g) *Se $OP = 7$ o programa deverá exibir todos os elementos da pilha sem desempilhá-los (`print`).*
   (h) *Se $OP = 8$ o programa deverá exibir todos os elementos da pilha na ordem reversa sem desempilhá-los (método criado na questão 9).*
   (i) *Se $OP = 0$ o programa deverá finalizar.*
   (j) *Se OP for qualquer outro valor, deverá ser ignorado.*
2. *Após a execução da operação determinada por OP, se $OP = 0$, então o programa deverá finalizar; caso contrário, o programa voltará a execução no passo 1.*

11. Crie uma FUNÇÃO que receba por parâmetro, uma string contendo uma expressão matemática que pode ter parênteses, colchetes e/ou chaves. A função deverá avaliar se todos os parênteses, colchetes e chaves foram corretamente abertos e fechados na expressão. Caso estejam corretamente abertos e fechados, a função deverá retornar `true`; caso contrário, deverá retornar `false`.

12. Crie uma FUNÇÃO que receba por parâmetro uma string e utilize uma pilha para verificar se essa string possui a forma $x\#y$, onde x é composto apenas pelas letras `A` e `B` e y é exatamente o inverso de x. A função deverá retornar `true` caso a string possua essa forma ou `false` caso contrário. Um exemplo de string com essa forma é `ABBABBB#BBBABBA`.

13. Crie uma FUNÇÃO que receba dois parâmetros: o primeiro é um número inteiro positivo na base 10; o segundo é outro número inteiro que represente uma base de destino. A função deverá imprimir na tela o número do primeiro parâmetro convertido para a base determinada no segundo parâmetro. Por exemplo: Se a função for chamada passando o valor 30 no primeiro parâmetro e 2 no segundo, deverá ser impresso 11110; se o valor for 30 e a base 8, deverá ser impresso 36; se o valor for 30 e a base 16 deverá ser impresso 1E; se o valor for 30 e a base 4, deverá ser impresso 132. Para bases maiores do que 10, utilize letras maiúsculas começando de `A` para suprir a limitada quantidade de caracteres numéricos.

14. Crie um PROGRAMA que implemente o jogo Torres de Hanói. Utilize três pilhas para representar os três pinos. Seu programa deverá obedecer a seguinte estrutura:

    1. *Programa lê do teclado um numero N cujo valor esteja entre 3 e 7. N será a quantidade de discos.*
    2. *Programa inicializa o jogo com os N discos no primeiro pino.*
    3. *Programa imprime uma visualização gráfica do estado dos três pinos apresentando onde estão os discos.*
    4. *Programa lê do teclado dois números inteiros: um indicando o pino de origem e outro indicando o pino de destino.*
    5. *Programa verifica se existe algum disco no pino de origem e se é possível realizar a movimentação do disco que está no topo desse pino para o pino de destino.*
       (a) *Se existir e for possível realizar o movimento, programa deverá realizar o movimento.*
       (b) *Se não for possível, programa mostra mensagem de erro e continua no passo 6.*
    6. *Programa verifica se todos os discos já estão no 3^o pino.*
       (a) *Se estiverem, programa finaliza com uma mensagem parabenizando o jogador.*
       (b) *Se não estiverem, programa volta a executar o passo 3.*

15. Crie uma FUNÇÃO que receba por parâmetro uma string que contenha uma expressão aritmética com representação in-fixa e converta essa expressão para a representação pós-fixa. A função deverá retornar uma string contendo essa expressão convertida.

16. Crie um PROGRAMA que leia do teclado uma expressão aritmética na forma in-fixa e realize o cálculo da expressão. Utilize a função criada na questão 15 para realizar a conversão e viabilizar o cálculo a partir da expressão na forma pós-fixa.