Universidade de Brasília Faculdade UnB Gama

Estruturas de Dados 1 - Gabarito da Prova Teórica 3

Nome:

Matrícula: Data:

Observações:

(A) A prova é individual e sem consulta, sendo vedado o uso de calculadoras e de telefones celulares.

(B) A interpretação dos comandos das questões faz parte da avaliação.

(C) Nas questões de múltipla escolha, apenas **uma** alternativa deve ser escolhida. Questões com **múltiplas marcações** ou **sem marcação** serão desconsideradas.

---

1. Considere o código abaixo:

```
1 template<typename T>
2 class List {
3     struct Node {
4         T info;
5         Node *prev, *next;
6     }
7
8     Node *head;
9 };
```

De acordo com as definições de listas encadeadas e duplamente encadeadas, podemos afirmar que a classe `List`

(A) não tem membros o suficiente para determinar o tipo de lista a ser implementada

(B) implementa uma lista simplesmente encadeada

(X) implementa uma lista duplamente encadeada

(D) não implementa uma lista

2. Em uma lista duplamente encadeada, o campo `prev` de `tail` aponta para

(A) `nullptr`

(B) o primeiro elemento, se existir

(X) o penúltimo elemento, se existir

(D) o último elemento, se existir

3. Considere as atribuições abaixo, feitas no contexto de uma lista duplamente encadeada com exatamente três elementos:

```
1 Node *p = tail->prev->prev;
2 Node *q = head->next->prev;
```

O campo `prev` de `q` aponta para qual elemento da lista?

(X) nenhum (nulo)

(B) terceiro

(C) segundo

(D) primeiro

As questões 04 e 05 se referem ao código abaixo, escrito em C++, no contexto de uma lista duplamente encadeada.

```
1 Node *node = new Node();
2 node->info = 0;
3 node->prev = tail;
4 node->next = nullptr;
5 tail ? tail->next = node : head = node;
6 tail = node;
```

4. O código acima implementa a

(A) inserção no início
(B) remoção do início
(X) inserção no final
(D) remoção do final

5. No código acima, qual troca de linhas interfere na correta execução do algoritmo?

(X) 5 e 6
(B) 3 e 4
(C) 2 e 5
(D) 2 e 4

6. Considere o código abaixo, que manipula nós de uma lista circular:

```
1 Node *node = list->tail;
2
3 for (int i = 0; i < 41; i++)
4     node = node->next;
```

Se a lista tem 7 elementos, ao final do laço o ponteiro `node` aponta para qual elemento da lista?

(A) nenhum (nulo)
(B) último
(X) penúltimo
(D) primeiro

7. Um tipo de dados abstrato é definido

(A) por sua interface e por sua implementação
(B) pela presença de métodos virtuais
(C) por sua implementação
(X) por sua interface

8. O tipo de dados abstrato onde o primeiro elemento a ser inserido é o último a ser removido é denominado

(A) lista auto-organizável
(B) lista circular
(X) pilha
(D) fila

9. Utilizando composição com uma lista simplesmente encadeada, a melhor implementação de uma fila delega as funções de inserção e remoção, respectivamente, para quais métodos da lista?

(A) `push_front()` e `pop_back()`
(B) `push_back()` e `pop_back()`
(X) `push_back()` e `pop_front()`
(D) `push_front()` e `pop_front()`

10. A busca das entradas `ACCACCBB` em uma lista auto-organizável, inicialmente vazia, cuja estratégia de organização é a ordenação estática ótima, resulta na lista

(A) `ABC`
(B) `BAC`
(C) `BCA`
(X) `CAB`

11. Considere o código abaixo:

```
1 priority_queue<double> pq { 3.2, -0.8, 1.3,
2     2.6, 5.4, 4.9, 7.0 };
3
4 for (int i = 0; i < 2; ++i)
5     pq.pop();
6
7 cout << pq.top() << '\n';
```

A linha 7 imprimirá qual valor?

(X) `4.9`
(B) `3.2`
(C) `2.6`
(D) `1.3`

12. Qual método não está presente na implementação da classe `forward_list` da STL do C++?

(A) `push_front()`
(B) `empty()`
(C) `clear()`
(X) `size()`

13. Qual das estruturas abaixo não tem uma implementação na STL do C++?

(X) lista circular
(B) lista com prioridades
(C) lista duplamente encadeada
(D) lista simplesmente encadeada

13. (3,5 pontos) Considere a implementação de uma lista duplamente encadeada abaixo:

```
1  #ifndef LIST_H
2  #define LIST_H
3
4  #include <ostream>
5  #include <initializer_list>
6
7  class List {
8      friend std::ostream& operator<<(std::ostream& os, const List& L);
9
10 public:
11     List();
12     List(const std::initializer_list<int>& elems);
13     ~List();
14
15     int front() const;
16     int back() const;
17
18     bool empty() const;
19     unsigned long size() const;
20
21     List& operator=(const List& L);
22     bool operator==(const List& L) const;
23
24     void push_front(int info);
25     void push_back(int info);
26
27     void pop_front();
28     void pop_back();
29
30     bool pop_last_but_one();
31
32 private:
33     struct Node {
34         int info;
35         Node *prev, *next;
36
37         Node(int i, Node *p, Node *n) : info(i), prev(p), next(n) {}
38     };
39
40     Node *head, *tail;
41     unsigned long _size;
42 };
43
44 #endif
```

Considere que todos os métodos, exceto o método `pop_last_but_one()`, estejam corretamente implementados.

O método `pop_last_but_one()` remove o penúltimo elemento da lista, se existir, retornando verdadeiro; ou retorna falso, caso a lista tenha menos do que dois elementos.

Implemente o método `pop_last_but_one()` com complexidade $O(1)$. A assinatura a ser utilizada é

```
bool List::pop_last_but_one();
```

A implementação deve conter, no máximo, 35 linhas. Não é necessário declarar nenhuma diretiva `include`. Use a próxima folha e siga as linhas conforme a numeração indicada. Escreva com letra legível, de preferência em letras de forma, e utilize um lápis. A implementação deve começar com a assinatura da função e deve terminar com o fim do bloco da função.

Solução: Usando os próprios métodos da classe:

```cpp
#include "list.h"

bool List::pop_last_but_one()
{
    if (size() < 2)
        return false;

    auto info = back();
    pop_back();
    pop_back();
    push_back(info);

    return true;
}
```

Solução que não utiliza os métodos da classe:

```cpp
#include "list.h"

bool List::pop_last_but_one()
{
    if (size() < 2)
        return false;

    auto prev = tail->prev->prev;

    tail->prev = prev;
    prev ? prev->next = tail : head = tail;
    _size--;

    return true;
}
```