Introdução

Considere o seguinte protótipo para a classe *List*, que implementa uma lista encadeada dinâmica:

```
typedef long ListEntry;

class List
{ public:
    List();
    ~List();
    bool Empty();
    bool Full();
    void Clear();
    long Size();
    void Insert(long p, ListEntry x);
    void Delete(long p, ListEntry &x);
    void Retrieve(long p, ListEntry &x);
    long Search(ListEntry x);
    string toString();
    string toStringAddr();
    long GetAddr(long p);

    bool Swap(ListEntry a, ListEntry b);

  private:
    // declaracao de tipos
    struct ListNode; // serah definido logo adiante no codigo

    typedef ListNode *ListPointer;
    struct ListNode
    { ListEntry Entry;          // tipo de dado colocado na lista
      ListPointer NextNode;     // ligacao para proximo elemento na lista
    };

    // campos
    ListPointer head;           // inicio da lista
    long count;                 // numero de elementos

    // metodos privados
    void SetPosition(long p, ListPointer &current);
};
```

Implemente o método `bool Swap(ListEntry a, ListEntry b)` que tem o seguinte comportamento:

i) Se a lista está vazia ou contém um único elemento ou $a=b$, o método não altera a lista e retorna *false*;
ii) Se a lista contém dois ou mais elementos
   a. Se os elementos *a* e *b* não forem encontrados na lista, o método não altera a lista e retorna *false*;
   b. Se os elementos *a* e *b* estão presentes na lista, o método troca os nós em que os elementos se encontram na lista de posição e retorna *true*;

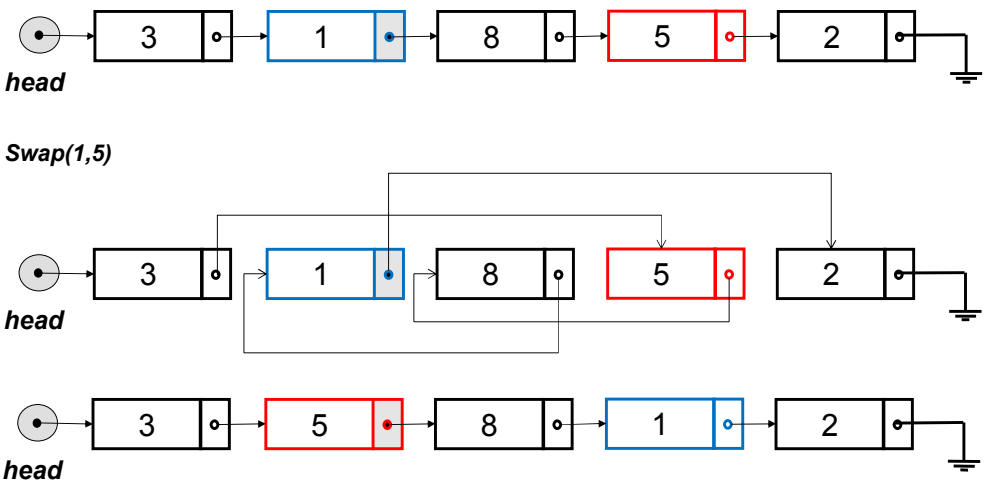
Atente que no caso (b) o método não deve trocar o valor existente campo de dados (Entry); o método deve reposicionar os ponteiros para que reflitam a troca de posição dos elementos.

Ao escrever o método, atente para os seguintes casos:

- *a* e *b* estão separados por um ou mais nós na lista,
- *a* e *b* estão adjacentes;
- *a* ou *b* é a cabeça ou o final da lista.

Assumindo que a lista tenha tamanho n, o método *Swap* deve realizar, no pior caso, $n+k_0$ operações (k_0 uma constante que não depende de n). Implementações realizando $k_1 \times n+k_0$ operações (k_0 e k_1 constantes independentes de n, com $k_1>1$) terão nota zero neste laboratório, mesmo que a avaliação automática realizada pela plataforma Web-CAT tenha resultado em nota acima de zero.

Na figura seguinte é mostrado um exemplo do comportamento do método *Swap*:



Exemplo de uso do método *Swap* da classe *List*:

```
int main()
{ List l;

  l.Insert(1,3);
  l.Insert(2,1);
  l.Insert(3,8);
  l.Insert(4,5);
  l.Insert(5,2);
  cout << "Antes Swap" << endl;
  cout << "Lista: " << l.toString() << endl;
  cout << "Nos..: " << l.toStringAddr() << endl;
  l.Swap(1,5);
  cout << "Apos Swap(1,5)" << endl;
  cout << "Lista: " << l.toString() << endl;
  cout << "Nos..: " << l.toStringAddr() << endl;

  return 0;
}
```

Saída:

```
Antes Swap
Lista: [3,1,8,5,2]
Nos..: [0x942eb0,0x942f18,0x942f28,0x942f38,0x942f48]
Apos Swap(1,5)
Lista: [3,5,8,1,2]
Nos..: [0x942eb0,0x942f38,0x942f28,0x942f18,0x942f48]
```

Note que o início da lista aparece mais à esquerda na saída e que os valores dos endereços dos nós também podem variar entre computadores ou entre execuções em um mesmo computador, por tratar-se de alocação dinâmica dos nós. O que deve ser notado é que, após a chamada ao métodos *Swap*, houve a troca entre os nós onde se encontravam os elementos 1 e 5.

Implementação

Você deve implementar o método Swap da classe List em C++, conforme o enunciado fornecido anteriormente. Utilize apenas programação estruturada. Juntamente com este documento, estão sendo disponibilizados os seguintes arquivos:

- *List.h*, *List.cpp* (interface e implementação parcial da classe *List*)
- *StudentEmptyTest.h* (arquivo de teste do aluno, no formato da plataforma CxxTest[1])

Você pode incluir quaisquer subalgoritmos (funções, procedimentos ou métodos) que se fizerem necessários nestes arquivos, porém não remova ou altere os métodos já fornecidos (caso contrário, é possível que a

[1] Você pode inserir mais casos de teste neste arquivo, caso queira tenha interesse em testar com mais detalhes seu código. Consulte http://cxxtest.com/guide.html#testAssertions.

plataforma CxxTest atribua pontuação menor que o máximo admissível ao seu trabalho, mesmo que ele esteja correto).

Submissão

Submeta sua implementação no sistema Web-CAT, disponível em http://kode.ffclrp.usp.br:8080/WebCat. Este laboratório deve ser submetido individualmente.

Antes de submeter os arquivos necessários, coloque seu nome completo em todos os arquivos sendo submetidos, na forma de comentário no início de cada arquivo (.h ou .cpp).

Compacte os seguintes arquivos em um único arquivo .zip (não utilize espaços no nome do arquivo compactado, nem adicione pastas/diretórios no arquivo compactado):

- *List.h*, *List.cpp* (interface e implementação da classe *List*)
- *StudentEmptyTest.h* (arquivo de teste do aluno, no formato da plataforma CxxTest)

Não inclua o programa principal, ou seja a função *main*(), na submissão ao Web-CAT. Respeite os nomes de arquivos, da classe e dos métodos. Submeta o arquivo compactado ao Web-CAT. Em caso de dúvida, procure o professor.

Avaliação

Na nota do trabalho, além dos critérios mencionados neste enunciado, também serão considerados os seguintes critérios (além daqueles já mencionados nas Disposições Gerais entregues no início do semestre):

- **Correção:** O programa faz o que foi solicitado? Faz tudo o que foi solicitado? Utiliza encapsulamento de informação? (i.e., acessa adequadamente os ADTs definidos?)
- **Eficiência: A**s operações são executadas da maneira mais eficiente para cada estrutura de dados? Evita código duplicado/redundante/não atingível?
- **Interface:** É simples de usar, genérico, prático, tolera os erros mais óbvios? O trabalho foi entregue dentro das especificações (um arquivo *.h* para cada *.cpp* implementando um ADT? Os arquivos estão em formato ZIP, com os nomes de arquivos solicitados)?
    - interface do programa;
    - implementação dos métodos;
- **Código fonte:** é claro (*layout*, espaçamento, organização em geral), nomes de variáveis são sugestivos, e há documentação/comentários apropriados no código? Faz uso de pré- e pós-condições? Quando aplicável, faz uso de sublagoritmos (funções, procedimentos ou métodos) adicionais que melhoram a legibilidade do(s) método(s) solicitado(s) sem comprometer sua eficiência (por exemplo, na notação assintótica **O(*n*),** onde ***n*** representa o tamanho da entrada?)