ARA7125 Estruturas de Dados I
Primeiro Semestre - 2015
Primeira Lista de Exercícios

1. Considere a seguinte definição de célula.

```
struct cel {
  pessoa * conteúdo;
  struct cel * seg;
};
typedef struct cel célula;
```

Note que o conteúdo de uma célula é um ponteiro para um tipo `pessoa`. Defina um tipo `pessoa` com pelo menos dois campos: um campo `char nome [MAX_NOME]`, onde `MAX_NOME` é uma constante que representa o número máximo de caracteres de um nome e um campo `int num`.

Considerando uma lista encadeada **sem** cabeça, faça as seguintes **funções na linguagem C**:

- `void Inserir (célula * p, pessoa * x);`
  - A função `Inserir` recebe o endereço de uma célula (`p`) e o endereço de uma pessoa (`x`), e insere um novo elemento entre a célula apontada por `p` e a seguinte.
- `pessoa * Remover (célula * p);`
  - A função `Remover` recebe o endereço de uma célula (`p`), remove o elemento que está na célula seguinte à apontada por `p`, e devolve o elemento removido. Você pode supor que existe pelo menos um elemento na lista.
- `celula * Buscar (célula * lst, char nome []);`
  - A função `Buscar` recebe o endereço da primeira célula de uma lista (`lst`) e uma cadeia de caracteres (`nome`), e devolve ou o endereço da primeira célula da lista que possui nome igual ao `nome`, ou `NULL`, caso não existir uma célula com tais características.

Você **deve** obedecer o protótipo das funções e lembre que tais funções **devem** considerar uma lista encadeada **sem** cabeça.

2. Faça um **programa em C** que use a lista encadeada sem cabeça do exercício anterior para inserir os presidentes da república que nomearam ministros para do Supremo Tribunal Federal e o número de ministros que

cada um nomeou. A lista está disponível em: `http://www.stf.jus.br/portal/ministro/ministro.asp?periodo=stf&tipo=quadro`.

Dica: Armazene o nome e os número de nomeações de cada presidente em um arquivo. Depois leia e insira cada presidente na lista.

a. Faça uma função que imprime o nome seguido do número de nomeaçõs de cada presidente que armazenado na lista começando pelo presidente armazenado na primeira célula da lista até o nome do presidente armazenado na última célula da lista. Use o seguinte protótipo: `void Imprime (célula * lst);`

b. Chame a função `Imprimir` para confirar se a sua lista armazena os nomes dos presidentes na ordem cronológica decrescente, isto é, primeiro deve aparecer a presidenta Dilma, depois o Lula, depois o Fernando Henrique, e assim por diante, até o presidente Manoel Deodoro da Fonseca. Você pode chamar a função `Imprime` para conferir isso. Se a sua lista não atende a essa restrição, remova todas as células dessa lista, e insira os presidentes na lista de tal forma que essa restrição seja atendida. Utilize as funções `Remover` e `Inserir` para remover todos os presidentes da lista e inserir os presidentes na ordem certa.

c. Imprima o número total de ministros nomeados por presidentes da ditadura militar. Use a função `Buscar` (definida no exercício anterior) para ter acesso às células desses presidentes.

d. Remova da lista os presidentes da ditadura militar. Use a função `Remova` definida no exercício anterior.

3. Considere a seguinte definição de célula.

```
struct cel {
  pessoa * conteúdo;
  struct cel * seg;
  struct cel * ant;
};
typedef struct cel célula;
```

Novamente, note que o conteúdo de uma célula é um ponteiro para um tipo `pessoa`. Defina um tipo `pessoa` com pelo menos dois campos: um campo `char nome [MAX_NOME]`, onde `MAX_NOME` é uma constante que representa o número máximo de caracteres de um nome e um campo `int num`.

Considerando uma lista duplamente encadeada **com** cabeça, faça as seguintes **funções na linguagem C**:

- `void Inserir (célula * p, pessoa * x);`

  - A função `Inserir` recebe o endereço de uma célula (`p`) e o endereço de uma pessoa (`x`), e insere um novo elemento entre a célula apontada por `p` e a seguinte.

- `pessoa * Remover (célula * p);`

  - A função `Remover` recebe o endereço de uma célula (`p`), **remove o elemento da célula apontada por** `p`, e devolve o elemento removido. Você pode supor que existe pelo menos um elemento na lista. Obs.: Note a diferença entre essa função e a função correspondente no exercício **1**.

- `celula * Buscar (célula * lst, char nome []);`

  - A função `Buscar` recebe o endereço da primeira célula de uma lista (`lst`) e uma cadeia de caracteres (`nome`), e devolve ou o endereço da primeira célula da lista que possui nome igual ao `nome`, ou `NULL`, caso não existir uma célula com tais características.

Você **deve** obedecer o protótipo das funções e lembre que, desta vez, tais funções **devem** considerar uma lista duplamente encadeada **com** cabeça.

4. Refaça o exercício **2**, considerando uma lista duplamente encadeada com cabeça e considerando as funções do exercício **3**.

5. Suponha que o conteúdo de uma célula de uma lista encadeada sem cabeça seja um número inteiro. Escreva uma função que recebe uma lista encadeada sem cabeça e devolve o endereço de uma célula que contém um conteúdo mínimo.

6. Defina e implemente as operações `Inserir`, `Remover` e `Buscar` usando uma **lista duplamente encadeada circular com cabeça**.

7. Suponha que o conteúdo de uma célula de uma lista é um número inteiro. Uma lista é crescente se o conteúdo de cada célula não é maior que o conteúdo da célula seguinte. Escreva uma função que recebe uma lista com cabeça e devolve 1 se a lista é crescente ou 0 caso contrário.

8. Escreva uma função que inverte uma dada **lista encadeada sem cabeça**. A lista resultante deve conter as mesmas células que a original, porém na ordem inversa. Não troque o conteúdo das células.

9. Escreva uma função que inverte uma dada **lista duplamente encadeada com cabeça**. A lista resultante deve conter as mesmas células que a original, porém na ordem inversa. Não troque o conteúdo das células.

10. Escreva uma função que recebe duas listas encadeadas com cabeça L_1 e L_2, e devolve uma nova lista com cabeça L_3 que corresponde à união das listas L_1 e L_2, isto é, $L_3 = L_1 \cup L_2$. Utilize em L_3 as células das listas L_1 e L_2.

11. Escreva uma função que recebe duas listas duplamente encadeadas **circulares** e com cabeça L_1 e L_2, e devolve uma nova lista duplamente encadeada circular com cabeça L_3 que corresponde à união das listas L_1 e L_2, isto é, $L_3 = L_1 \cup L_2$. Utilize em L_3 as células das listas L_1 e L_2. Está função **deve ser** muito mais rápida que a função do exercício **9**.

12. Escreva uma função que recebe duas listas encadeadas com cabeça L_1 e L_2, e devolve uma nova lista com cabeça L_3 que corresponde à intersecção das listas L_1 e L_2, isto é, $L_3 = L_1 \cap L_2$. Utilize em L_3 as células das listas L_1 e L_2.

13. Escreva uma função que recebe duas listas encadeadas com cabeça L_1 e L_2, e devolve uma nova lista com cabeça L_3 que corresponde à diferença entre os elementos da lista L_1 com os elementos da lista L_2, isto é, $L_3 = L_1 - L_2$. Utilize em L_3 as células das listas L_1 e L_2.