Universidade Tecnológica Federal do Paraná (UTFPR)
Departamento Acadêmico de Informática (DAINF)
Estrutura de Dados I
Professor: Rodrigo Minetto
Exercícios (lista encadeada)

Exercícios (seleção): necessário entregar TODOS (moodle)!

Exercício 1) Escreva uma função que recebe uma lista encadeada como entrada e retorna a soma dos elementos. Por exemplo, se lista = $\{5, 6, 7, 8, 9, 4, 3, 2, 1, 0\}$ então o retorno da função é 45. A função deve utilizar o seguinte protótipo:

```
int sum (List *l);
```

Para resolver este exercício utilize o código **sum.c** (em '**arquivos.zip**') ou alternativamente codifique em **sum-codeblocks.c**.

Exercício 2) Escreva uma função para inserir na cauda da lista (última posição). Observe que o ponteiro na função main aponta para a cabeça da lista, ou seja, é necessário percorrer a lista para determinar a posição correta. Considere o seguinte protótipo para a sua função:

```
List* insert_back (List *l, int elem);
```

Para testar o seu código utilize no exercício **sum.c** a função **insert_back** (ao invés da **insert**), o que deve produzir como saída o seguinte:

```
List: 0 1 2 3 4 9 8 7 6 5  <-  inserção com o insert_back
Sum = 45
```

Qual a complexidade dessa operação em termos da notação $\mathcal{O}(?)$

Para resolver este exercício codifique a função **insert_back** em **list.c** (em '**arquivos.zip**') ou alternativamente codifique em **sum-codeblocks.c**.

Exercício 3) Escreva uma função que recebe uma lista encadeada e um inteiro k como entrada e retorna verdadeiro se o elemento k pertence a lista ou falso caso contrário. A função deve utilizar o seguinte protótipo:

```
int in (List *l, int k);
```

Para resolver este exercício utilize o código **in.c** (em '**arquivos.zip**') ou alternativamente codifique em **in-codeblocks.c**.

Exercício 4) Escreva uma função que recebe duas listas encadeadas A e B ordenadas como entrada e faz a fusão de A e B, mantendo-a ordenada. Não devem ser alocados (criados) nós extras. Os nós serão religados para compor a nova lista C ordenada. Por exemplo, se $A = \{0, 2, 4, 6, 8\}$ e $B = \{1, 3, 3, 5\}$, então $C = \{0, 1, 2, 3, 3, 4, 5, 6, 8\}$, tal que C tem os nós de A e B, então para desalocar memória basta liberar C. A função deve utilizar o seguinte protótipo:

```
List* merge (List *A, List *B);
```

Para resolver este exercício utilize o código **merge.c** (em '**arquivos.zip**') ou alternativamente codifique em **merge-codeblocks.c**.

Exercícios (aprofundamento): não é necessário entregar mas é importante estudar!

Exercício 5) Escreva uma função que recebe uma lista encadeada como entrada e retorna o tamanho da lista. Por exemplo, se lista = $\{5, 6, 7, 8, 9, 4, 3, 2, 1, 0\}$ então o retorno da função é 10. A função deve utilizar o seguinte protótipo:

```
int size (List *l);
```

A estrutura de dados e operações básicas estão em anexo ao material da aula (**arquivos.zip**) para auxiliar.

Exercício 6) Escreva uma função que recebe uma lista encadeada como entrada e retorna o maior elemento armazenado. Por exemplo, se lista = $\{5, 6, 7, 8, 9, 4, 3, 2, 1, 0\}$ então o retorno da função é 9. A função deve utilizar o seguinte protótipo:

```
int max (List *l);
```

Exercício 7) Escreva uma função que recebe duas listas encadeadas A e B como entrada e retorna true se elas são iguais, ou falso caso contrário. Para duas listas serem iguais elas precisam ter os elementos na mesma ordem e com os mesmos valores. A função deve utilizar o seguinte protótipo:

```
int similar (List *A, List *B);
```

Exercício 8) Escreva uma função que recebe uma lista encadeada A como entrada e realiza uma cópia, ou seja, para cada nó em A realize a alocação de um novo nó em uma nova lista B. Para ter certeza que a cópia foi realizada de forma correta, destrua a lista A e após imprima o conteúdo de B. A função deve utilizar o seguinte protótipo:

```
List* copy (List *A);
```

Exercício 9) Escreva uma função que recebe duas listas encadeadas A e B como entrada e cria um novo conjunto C de intersecção entre elas. Por exemplo, se $A = \{0, 5, 10, 15, 20, 25, 30\}$ e $B = \{0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30\}$, então o conjunto intersecção é $C = \{0, 15, 30\}$. A função deve utilizar o seguinte protótipo:

```
List* intersection (List *A, List *B);
```

Qual a complexidade em notação $\mathcal{O}(?)$ dessa operação?

Exercício 10) Escreva uma função que recebe uma lista encadeada e um inteiro k como entrada e insere k na lista respeitando a ordenação. Por exemplo, se os valores de k sucessivos são $9, 0, 4, 2, 7$, então $l = \{0, 2, 4, 7, 9\}$. A função deve utilizar o seguinte protótipo:

```
List* insert_sort (List *l, int k);
```

Exercício 11) Escreva uma função que recebe uma lista encadeada como entrada e imprime a lista em ordem reversa. Por exemplo, se lista $= \{1, 2, 3, 4, 5\}$ então o retorno da função é $\{5, 4, 3, 2, 1\}$. A função deve utilizar o seguinte protótipo:

```
void print_reverse (List *l);
```

Ps. não use vetores apenas listas (se precisar pode criar ou alocar uma nova lista com o mesmo tamanho).

Exercício 12) Listas encadeadas podem ser projetadas para representar polinômios. Por exemplo, cada nó da lista pode armazenar o coeficiente e expoente de um termo do polinômio (em ordem decrescente de grau) e um ponteiro para o próximo termo do polinômio; assim o polinômio $P = 3x^5 - 4x^2$ pode ser representado em uma lista encadeado por

```
      ------------      -------------
P -> |c = 3, e = 5| -> |c = -4, e = 2| -> ||
      ------------      -------------
```

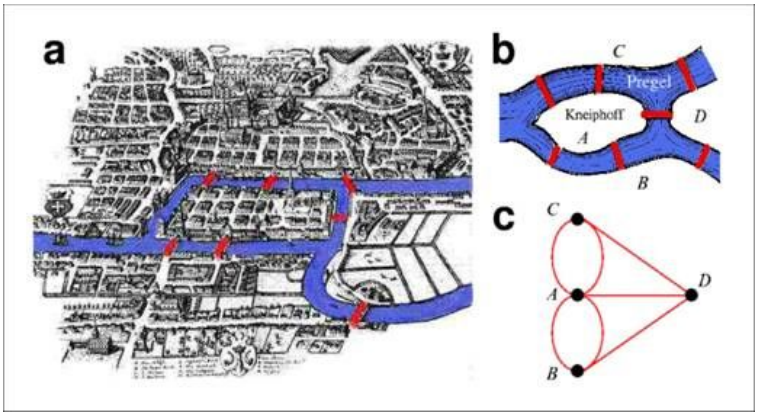
Escreva uma função tal que dado dois polinômios cria um novo polinômio com o resultado da soma deles. Por exemplo:

```
Polinômio A: 7^5 + 5^2 + 4^1 + 2^0
Polinômio B: 3^6 + 5^2 - 4^1 - 5^0
Soma: 3^6 + 7^5 + 10^2 - 3^0
```

A função deve utilizar o seguinte protótipo:

```
List* poly_sum (List *A, List *B);
```

Exercício 13) No início do século XVIII especula-se que os cidadãos da cidade russa de Königsberg costumavam passar suas tardes de domingo a caminhar em torno da sua localidade. Königsberg naquela época era constituída por quatro áreas de terra separadas pelo rio pregel, sobre o qual haviam sete pontes, tal como ilustrado na figura abaixo (letras A e B). O problema que os cidadãos fixaram era caminhar ao redor da cidade, cruzando cada uma das sete pontes apenas uma vez e, se possível, retornar ao seu ponto de partida. A partir do problema das pontes, Leonhard Euler sistematizou um novo campo da matemática — era o surgimento da **teoria dos grafos**. O matemático não precisou de mais de uma quinzena de dias para resolver o enigma. Para isso, ele criou um modelo matemático que simulasse a cidade russa, o que chamamos hoje de **grafo**. Durante a elaboração do



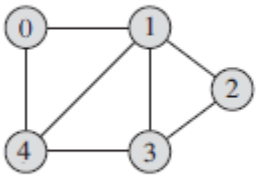
grafo, ele representou as porções de terra (ilhas e margens) por pontos (vértices) e as pontes por linhas (arestas) ligando esses pontos (veja o esquema de Euler na figura abaixo, letra C).

A representação computacional de um grafo, de forma geral (mas não exclusiva), segue duas abordagens clássicas:

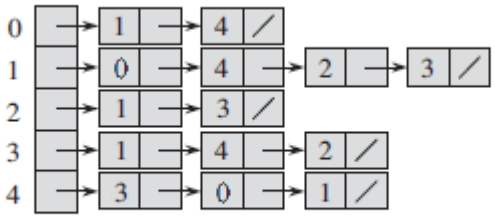
- matriz de adjacências
- listas de adjacências

Cada uma das representações tem suas vantagens e suas desvantagens.

A matriz de adjacências de um grafo é uma matriz booleana com colunas e linhas indexadas pelos vértices, tal que o valor **um** indica que existe uma aresta conectando os respectivos vértices e **zero** quando não há conexão direta. Ao contrário de uma matriz, uma lista de adjacência não possui zeros e uns para representar as conexões dos vértices de um grafo, elas formam uma versão mais compacta através do uso de uma lista encadeada, onde em cada posição desta lista existe uma outra lista encadeada. Cada lista individual mostra a quais nós um dado nó é adjacente. A figura abaixo mostra um grafo qualquer em (a), a lista de adjacências correspondente (b) e a respectiva matriz de adjacência (c).



(a)



(b)

	0	1	2	3	4
0	0	1	0	0	1
1	1	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	1
4	1	1	0	1	0

(c)

Neste exercício sua tarefa é codificar um grafo atráves de listas de adjacências (que utilizam listas encadeadas). Para tanto, use a estrutura definida em '**graph.h**' e implemente as funções necessárias em '**graph.c**' (em '**arquivos.zip**'). O programa '**test-graph.c**' testa a funcionalidade do código ao construir e imprimir as arestas referente ao grafo acima apresentado. Considere seu código correto se a saída for:

```
0 -> 1 4
1 -> 4 3 2 0
2 -> 3 1
```

```
3 -> 4 2 1
4 -> 3 1 0
# vertices = 5, # arestas = 7
```