---

1. Resolva os problemas a seguir:
   (a) O Algoritmo A usa $8n$ operações básicas, enquanto o algoritmo B usa n^2 operações básicas. Determine o valor n_0 tal que A é melhor do que B para todo $n \geq n_0$.
   (b) O Algoritmo A usa $10n \log_2 n$ operações básicas, enquanto o algoritmo B usa n^2 operações básicas. Determine o valor n_0 tal que A é melhor do que B para todo $n \geq n_0$.

2. Ordene a seguinte lista de funções de acordo com sua complexidade assintótica

$6n \log_2 n$	2^{100}	$\log_2 \log_2 n$	$\log_2^2 n$	$2^{\log_2 n}$	2^{2n}	$\sqrt{n}$	$n^{0.01}$	$1/n$	$4n^{3/2}$
$3n^{0.5}$	$5n$	$2n \log_2^2 n$	2^n	$n \log_4 n$	4^n	n^3	$n^2 \log_2 n$	$4^{\log_2 n}$	$\sqrt{\log_2 n}$

3. Implemente uma lista duplamente encadeada com nós do tipo `struct node`, contendo três campos `int ele`, `struct node* prev` e `struct node* next`. Implemente as operações básicas:

   - `front()` que retorna um ponteiro para o primeiro elemento da lista.
   - `insert(int e)` para inserir um inteiro `e` na lista
   - `remove(int e)` para remover um inteiro `e` da lista (caso o inteiro `e` não esteja presente na lista, a função deve retornar um ponteiro nulo)
   - `insertBefore(int p,int e)` procura um inteiro `p` presente na lista e insere um inteiro `e` antes dele (caso o inteiro `p` não esteja presente na lista, a função deve retornar um ponteiro nulo).
   - `insertLast(e)` insere um inteiro `e` na última posição da lista.

4. Suponha que você tem dois números representados por duas listas encadeadas, onde cada nó contem um dígito. Os dígitos são armazenados em ordem inversa, de modo que o primeiro dígito de um número encontra-se na cabeça da lista. Implemente uma função que adiciona dois números e retorna a soma como uma lista encadeada.

   Exemplo:

   Entrada: $(3 -> 1 -> 5) + (5 -> 9 -> 2)$

   Saída: $8 -> 0 -> 8$

5. Modifique o pseudo-código do algoritmo *quickselect* de modo que ele retorne os k menores elementos.

6. Existem dois tipos de lutadores de luta-livre: “os bons” e “os malvados”. Entre qualquer par de lutadores, pode ou não haver uma rivalidade. Suponha que temos n lutadores e que temos uma lista de r pares de lutadores para os quais existem rivalidade. Forneça um algoritmo de tempo $O(n+r)$ que determina se é possível classificar os lutadores como bons ou malvados de modo que cada rivalidade seja entre um “bom” e outro “malvado”. Se tal classificação é possível, o seu algoritmo deve fornecê-la.