Obs. Nos exercícios com listas ligadas, vamos supor que temos

```
typedef struct node *link;
struct node { Item item; link next; }
```

1. [3.36 do Sedgewick] Escreva uma função que rearranja os elementos de uma lista dada, pondo todos os elementos que ocorrem nas posições pares antes de todos os elementos de ocorrem nas posições ímpares. Você deve preservar a ordem relativa dos elementos pares entre si, e a ordem relativa dos elementos ímpares entre si.

2. [4.6 do Sedgewick] Este é um exercício sobre pilhas. Na seqüência

```
E A S * Y * Q U E * * * S T * * * I O * N * * *
```

uma letra significa *empilhe* e um asterisco *desempilhe*. Qual é a seqüência de letras devolvida pelos *desempilhe*?

3. [4.7 do Sedgewick] Continuemos usando a notação do exercício anterior. Diga como devemos inserir asteriscos na seqüência `E A S Y` para que os *desempilhe* devolva as seguintes seqüências: (*i*) `E A S Y`, (*ii*) `Y S A E`, (*iii*) `A S Y E`, (*iv*) `A Y S E`, (*v*) `E Y A S`. Note que nem sempre podemos obter a saída desejada; nestes casos, argumente por que a dada saída não pode ser obtida desta forma.

4. [4.9 do Sedgewick] Converta a expressão

```
( 5 * ( ( 9 * 8 ) + ( 7 * ( 4 + 6 ) ) ) )
```

para a notação pósfixa e para a notação infixa.

5. [4.31 do Sedgewick] Este é um exercício sobre filas. Na seqüência

```
E A S * Y * Q U E * * * S T * * * I O * N * * *
```

uma letra significa inserção no fim da fila e um asterisco significa remoção do começo da fila. Qual é a seqüência de letras devolvida por esta seqüência de operações em nossa fila?

6. [O problema das panquecas] Suponha que temos, por exemplo, a seqüencia de números

```
8 2 3 5 1 4 9 7 6 0
```

Imagine que esta seqüência indica uma pilha de panquecas de 10 tamanhos distintos, empilhadas de forma que a menor está no fundo da pilha, a maior é a quarta panqueca a partir do fundo da pilha, a segunda maior está no topo, etc. As únicas operações permitidas são inverter um segmento inicial desta seqüência. Portanto, existem 9 operações não-triviais para a pilha acima; uma delas produz, por exemplo,

```
5 3 2 8 1 4 9 7 6 0
```

Podemos chamar esta operação deste exemplo de 'operação número 4', já que invertemos a ordem das 4 panquecas do topo. O problema é o seguinte: dados um inteiro positivo `n` e uma permutação dos inteiros não-negativos menores do que `n`, determinar uma seqüência de operações (permitidas) que resultam na seqüência

```
0 1 2 ... n - 1
```

Isto é, com as panquecas em ordem crescente, do topo para o fundo. Escreva uma função de protótipo

```
void imprima_solucao(int n, int panq[]);
```

para resolver este problema.

7. [5.11 do Sedgewick] Escreva um programa recursivo que converte uma expressão em notação infixa em notação pósfixa.

8. [5.14 do Sedgewick] Escreva uma função recursiva que remove o último elemento de uma lista ligada.

9. [5.17 do Sedgewick] Escreva uma função recursiva de protótipo

```
Item max_el(link h);
```

que recebe um ponteiro para o começo de uma lista ligada (sem cabeça e sem cauda) e que devolve a maior chave (isto é, campo `item`) presente na lista, onde assumimos que `Item` é o tipo `int`.

10. Considere a seguinte função de hashing:

```
int hash(char *v, int M)
  { int h = 0, a = 128;
    for (; *v != '\0'; v++) h = (a*h + *v) % M;
    return h;
  }
```

Suponha que o tamanho `M` da tabela que está sendo usada é `1024`. O usuário percebe que muitas colisões ocorrem quando esta função de hashing é utilizada para processar as palavras que ocorrem em um dado texto (isto é, o espalhamento esperado não ocorre). Você teria uma explicação para este fenômeno? Em particular, quando ocorre desta função devolver o mesmo valor para chaves distintas?

11. [14.16 de Sedgewick] Suponha que usamos uma tabela de hashing com resolução de colisões por encadeamento. Suponha que inserimos `N` itens em nossa tabela, que está inicialmente vazia. No pior caso (isto é, tendo muito azar com a função de hashing), quanto tempo pode demorar este processo? Suponha agora que resolvemos manter as nossas listas em ordem crescente de chaves; quanto tempo pode demorar este processo? (Nesta questão, estamos interessados em saber se o tempo é proporcional a `N`, proporcional a $\mathtt{N}^2$, proporcional a $\mathtt{N} \log \mathtt{N}$, etc).

12. [14.17 de Sedgewick] Suponha que estamos utilizando uma tabela de hashing com `M` entradas, com resolução de colisões por encadeamento (as `M` listas não são mantidas em ordem). Suponha que usamos a função de hashing que devolve $11k \bmod \mathtt{M}$ quando a entrada é a k-ésima letra do alfabeto (por exemplo, `C` corresponde a $k = 3$). Suponha que inserimos em uma tabela iniclamente vazia, nesta ordem, as chaves

```
E A S Y Q U T I O N
```

onde $\mathtt{M} = 5$. Desenhe diagramas que ilustram este processo de inserção.

13. [14.18 de Sedgewick] Continuando com a questão anterior, desenhe a configuração final da tabela agora supondo que mantemos as nossas listas em ordem crescente das chaves. A sua resposta depende da ordem de inserção das chaves?

14. [9.1 de Sedgewick] Este é um exercício sobre filas de prioridade. Na seqüência

```
P R I O * R * * I * T * Y * * * Q U E * * * U * E
```

uma letra significa a inserção daquela letra em uma fila e um asterisco significa a remoção do item de maior prioridade desta fila. Qual é a seqüência de chaves devolvida pelas 12 operações de remoção acima?

15. [9.3 e 9.4 de Sedgewick] Explique como usar uma fila de prioridades para implementar uma pilha. Explique como usar uma fila de prioridades para implementar uma fila.

16. [9.22 de Sedgewick] Suponha que a fila de prioridades da Questão 14 é implementada usando-se heaps. Desenhe os 25 heaps resultantes após cada uma das 25 operações daquela questão.

17. O algoritmo heapsort demora tempo $O(n \log n)$ para ordenar um vetor de n itens. Dê um argumento preciso para justificar esta asserção.

18. Esta questão baseia-se na rotina de partição do quicksort. Considere o seguinte código:

```
select(Item a[], int l, int r, int k)
  { int i;
    if (r <= l) return;
    i = partition(a, l, r);
    if (i > k) select(a, l, i-1, k);
    if (i < k) select(a, i+1, r, k);
  }
```

Qual é o efeito da chamada `selection(a, 0, N-1, N/2)`?

19. Considere o seguinte código:

```
select(Item a[], int l, int r, int k)
{ while (r > l) {
    int i = partition(a, l, r);
    if (i >= k) r = i-1;
    if (i <= k) l = i+1;
  }
}
```

Qual é o efeito da chamada `selection(a, 0, N-1, N/2)`?