Obs. Nas questões abaixo, suporemos que temos

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

1. [**3** pontos] Suponha que estamos utilizando uma tabela de hashing com `M` entradas, com resolução de colisões por encadeamento, com as `M` listas mantidas em ordem alfabética. Suponha que usamos a função de hashing que devolve $13k \bmod \texttt{M}$ quando a entrada é a k-ésima letra do alfabeto (por exemplo, `C` corresponde a $k = 3$ e `Z` a $k = 26$). Suponha que inserimos em uma tabela inicialmente vazia, nesta ordem, as chaves

```
Q U E S T A O F C I L D
```

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

2. [**3** pontos] Considere as seguintes funções (convenção: listas sem cabeça e sem cauda):

```
link merge(link a, link b)
  { struct node head; link c = &head;
    <imprima as listas dadas por a e b>
    while ((a != NULL) && (b != NULL))
      if (less(a->item, b->item)) { c->next = a; c = a; a = a->next; }
      else { c->next = b; c = b; b = b->next; }
    if (a == NULL) c->next = b; else c->next = a;
    <imprima a lista dada por head.next>
    return head.next;
  }
link sort(link c)
  { link a, b;
    if (c->next == NULL) return c;
    a = c; b = c->next;
    while ((b != NULL) && (b->next != NULL))
      { c = c->next; b = b->next->next; }
    b = c->next; c->next = NULL;
    return merge(sort(a), sort(b));
  }
```

Simule a execução da chamada `a = sort(a)`, quando a lista `a` contém 5 células, com os seguintes conteúdos:

```
333 111 444 111 555
```

Nesta questão, você deve dizer claramente qual é a saída.

3. [**4** pontos] Descreva *cuidadosamente* o projeto* de um programa que resolve o seguinte problema: (*i*) a entrada de seu programa é formada por um inteiro k e um texto T e (*ii*) a saída de seu programa deve ser a lista das k palavras mais freqüentes no texto, com suas respectivas freqüências (número de ocorrências em T). O seu projeto deve supor que a entrada T pode ser grande; por exemplo, T poderia ser um livro como *Anna Karenina* ($\sim$ 350.000 palavras).

Qual seria a *complexidade de tempo* de um programa que fosse implementado seguindo-se seu projeto? Isto é, se T tem n palavras, quanto tempo levaria seu programa para processar o par (k, T)? Responda a mesma pergunta para a *complexidade de espaço* de um tal programa (quanto espaço ele usa).

Observação. Você deve argumentar em detalhe por que o programa teria estas propriedades (em particular, você deve explicitar as hipóteses que você usa em sua análise).

---

* Dê os protótipos das funções principais, com uma descrição do que elas devem fazer e como elas devem ser implementadas (não esqueça do `main()`).