1. Considere um tipo de mapeamento que associa números de RG (chaves) a nomes de pessoas (valores), como segue:

```
typedef struct map {
   int rg;
   char nome[31];
   struct map *prox;
} *Map;
```

Com base nessa definição de mapeamento:

(*a*) Crie a função `Map no_map(int r, char n[], Map p)`, que cria um nó do mapeamento devidamente preenchido (com os valores `r`, `n` e `p`) e, no final, devolve seu endereço como resposta.

(*b*) Crie a função recursiva `void insm(int r, char n[], Map *M)`, que insere o par (`r`,`n`) no mapeamento apontado indiretamente pelo ponteiro `M`. A função deve garantir que o par seja inserido corretamente, de modo que a ordenação da lista seja mantida, e que não hajam chaves repetidas.

(*c*) Crie a função recursiva `void remm(int r, Map *M)`, que remove do mapeamento apontado indiretamente pelo ponteiro `M` o nó que contém a chave `r`.

(*d*) Crie a função recursiva `char *nome(int r, Map *M)`, que devolve o nome associado ao número de RG indicado pelo parâmetro `r`, no mapeamento apontado indiretamente pelo ponteiro `M`. Caso a chave `r` não esteja no mapeamento, a função deve devolver uma cadeia de caracteres vazia.

(*e*) Crie a função recursiva `void exibem(Map M)`, que exibe no vídeo todos os pares (`r`,`n`) existentes no mapeamento apontado diretamente pelo ponteiro `M`.

(*f*) Crie um programa para testar as funções criadas nos itens anteriores.

2. Considere um tipo de mapeamento que associa títulos de livros (chaves) a nomes de autores (valores), como segue:

```
typedef struct map {
   char titulo[42];
   char autor[31];
   struct map *prox;
} *Map;
```

Com base nessa definição de mapeamento[1]:

(*a*) Crie a função `Map no_map(char t[], char a[], Map p)`, que cria um nó do mapeamento devidamente preenchido (com os valores `t`, `a` e `p`) e, no final, devolve seu endereço como resposta.

(*b*) Crie a função recursiva `void insm(char t[], char a[], Map *M)`, que insere o par (`t`,`a`) no mapeamento apontado indiretamente pelo ponteiro `M`. A função deve garantir que o par seja inserido corretamente, de modo que a ordenação da lista seja mantida, e que não hajam chaves repetidas.

(*c*) Crie a função recursiva `void remm(char t[], Map *M)`, que remove do mapeamento apontado indiretamente pelo ponteiro `M` o nó que contém a chave `t`.

(*d*) Crie a função recursiva `char *autor(char t[], Map *M)`, que devolve o nome de autor associado ao título de livro indicado pelo parâmetro `t`, no mapeamento apontado indiretamente pelo ponteiro `M`. Caso a chave `t` não esteja no mapeamento, a função deve devolver uma cadeia de caracteres vazia.

(*e*) Crie a função recursiva `void exibem(Map M)`, que exibe no vídeo todos os pares (`t`,`a`) existentes no mapeamento apontado diretamente pelo ponteiro `M`.

(*f*) Crie um programa para testar as funções criadas nos itens anteriores.

[1] *Use as funções* `strcpy()` *e* `_stricmp()`*, definidas no arquivo* `string.h`*, para copiar e comparar cadeias de caracteres.*