

Estudante:	Matrícula:
-------------------	-------------------

Instruções:

- Este exame possui 2 página(s). Verifique se sua cópia do exame está completa.
- Esta prova é sem consulta:** você não tem permissão para consultar o livro texto, suas notas de aula, a Internet, seus colegas ou quaisquer outras pessoas ou fontes para concluir o exame.
- Se você acredita que alguma pergunta esteja subespecificada, anote as suposições que você teve que fazer para chegar a sua resposta e justifique-as como parte de sua resposta à pergunta.
- Lembre-se de indentar o código** de maneira apropriada e atente-se a organização, clareza e legibilidade do código!

Utilize o espaço a seguir, referente ao arquivo de cabeçalho **abin.h**, para a definição das estruturas solicitadas na primeira questão (se atente ao nome dos campos!). Note que o arquivo de cabeçalho já contém a declaração dos protótipos das funções a serem implementadas nas questões posteriores.

abin.h

```
1 #ifndef ABIN_H_
2 #define ABIN_H_
3 #include <stdio.h>
4 #include <stdbool.h>
5 typedef struct Funcionario {
6
7
8
9 } Funcionario;
10
11 typedef struct Empregados {
12
13
14 }
```

abin.h (continuação)

```
14 char *le_linha(FILE *p);
15 Funcionario le_funcionario(char *linha);
16 Empregados *le_empregados(char *arquivo);
17
18 // Retorna true se os 'k' primeiros
19 // funcionarios estao ordenados em ordem
20 // decrescente ou false caso contrario.
21 bool ordenado(Empregados e, int k);
22
23
24
25
26
27 #endif
```

- (3 pontos) A ABIN (Agência Brasileira de Inteligência) precisa processar um arquivo contendo seus funcionários, para produzir uma lista com os mais qualificados. Cada linha deste arquivo armazena o registro de um empregado, contendo sua matrícula, nome e quantidade de cursos de especialização. Segue, abaixo, um exemplo deste arquivo:

abin.txt

```
1 231323; Bill Gates; 3
2 231324; Paul Allen; 2
3 231325; Lery Page; 2
```

- (1,5 pontos) Criar a estrutura para armazenar um **Funcionário**. Sua estrutura deve conter os seguintes campos: **matricula** (string), **nome** (string) e **num_especializacoes** (inteiro não negativo).
 - (1,5 pontos) Criar a estrutura **Empregados** para guardar todos os funcionários da empresa. Sua estrutura deve conter os seguintes campos: **empregados** (arranjo de Funcionario) e **n** (inteiro; número de funcionários).
- (4 pontos) Crie uma função **le_linha**, seguindo o protótipo especificado no arquivo de cabeçalho (página 1), para ler uma linha e retorná-la como uma string. Cada linha pode conter no máximo 150 caracteres. **Dicas:** lembre-se de considerar o **\0** e lembre-se de alocar a string **dinamicamente**.
 - (6 pontos) Complete a função **le_funcionario** a seguir, que recebe uma string que corresponde a uma linha do arquivo, e retorna os dados armazenados em um objeto da estrutura **Funcionario** descrita acima.

abin.c

```

1 #include "abin.h"
2 #include <stdlib.h>
3 Funcionario le_funcionario(char *linha) {
4     Funcionario f; int i;
5     // Aloque, inicialmente, espaco para 1 caractere no nome e matricula:
6     f.matricula = _____;
7     f.nome = _____;
8     f.matricula[0] = '\0'; f.nome[0] = '\0';
9     for (i = 0; linha[i] != _____; i++) { // Ate onde vai a matricula?
10        // Aumente o espaco da matricula para caber mais um caractere:
11        f.matricula = _____;
12        f.matricula[i] = linha[i];
13        f.matricula[i + 1] = '\0';
14    }
15    // 'i' sai do loop anterior como sendo o indice do delimitador.
16    int j = 0;
17    for (i = i + 1; linha[i] != _____; i++, j++) { // Ate onde vai o nome?
18        // Aumente o espaco do nome para caber mais um caractere:
19        f.nome = _____;
20        f.nome[j] = linha[i];
21        f.nome[j + 1] = '\0';
22    }
23    int mult = 1; f.num_especializacoes = 0;
24    // 'i' sai do loop anterior como sendo o indice do delimitador.
25    for (int j = strlen(linha) - 1; j > i; j--) {
26        f.num_especializacoes += (_____) * mult; // Converta os caracteres
27        mult *= 10;                               // para inteiro
28    }
29    return f;
30 }

```

4. (6 pontos) Complete a função **le_empregados** a seguir, que recebe o nome de um arquivo, processa este arquivo, e retorna um ponteiro para **Empregados** (nulo em caso de erro) com todos os funcionários presentes no arquivo.

abin.c

```

31 Empregados *le_empregados(char *arquivo) {
32     FILE *p = _____; // Abra o arquivo de texto p/ leitura.
33     if (_____) return NULL; // E se algo errado acontecer?
34     // Aloque espaco para UM valor do tipo Empregados.
35     Empregados *e = _____;
36     e->empregados = NULL; // Inicialmente nao temos nenhum empregado.
37     e->n = 0;
38     char *linha = le_linha(p);
39     while (linha != NULL) {
40         Funcionario f = le_funcionario(linha);
41         // Abra espaco no arranjo 'empregados' para um novo funcionario:
42         e->n = e->n + 1;
43         int bytes = e->n * _____;
44         e->empregados = _____;
45         e->empregados[e->n - 1] = f;
46         linha = le_linha(p);
47     }
48     _____; return e; // Feche o arquivo
49 }

```

5. (6 pontos) Crie uma função **RECURSIVA** chamada **ordenado** que recebe um parâmetro do tipo **Empregados** e outro parâmetro inteiro k . Sua função deve retornar verdadeiro se os k primeiros empregados estão em ordem **decrecente**, de acordo com o número de especializações. Você pode implementar funções auxiliares, que também devem ser **RECURSIVAS**; isto é, **nenhuma função pode conter laços** como **for**, **while** ou **do-while**.