

<b>Estudante:</b>	<b>Matrícula:</b>
-------------------	-------------------

**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 **turma.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.

**turma.h**

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

**turma.h (continuação)**

```
14 char *le_linha(FILE *p);
15 Aluno le_aluno(char *linha);
16 Turma *le_turma(char *arquivo);
17 // Retorna o indice a partir do qual
18 // todos os alunos possuem nota >= k.
19 // Assuma que os alunos estao ordenados
20 // de forma crescente. Exemplo de chamada
21 // da funcao: indice(t, 0, 70)
22 bool indice_k(
23     Turma t,
24     unsigned int i,
25     unsigned int k
26 );
27 #endif
```

- (3 pontos) Os dados dos alunos de PC são armazenados em um arquivo **turma.txt**. Cada linha deste arquivo armazena o registro de um aluno, contendo sua matrícula, nome e nota final. Segue um exemplo deste arquivo:

**turma.txt**

```
1 18791955# Albert Einstein#60
2 16431727# Isaac Newton#87
3 18671934# Marie Curie#90
```

- (1,5 pontos) Criar a estrutura para armazenar um **Aluno**. Sua estrutura deve conter os seguintes campos: **matricula** (string), **nome** (string) e **nota** (inteiro não negativo).
  - (1,5 pontos) Criar a estrutura **Turma** para guardar todos os alunos da turma. Sua estrutura deve conter os seguintes campos: **alunos** (arranjo de **Aluno**) e **n** (inteiro; número de alunos).
- (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\_aluno** a seguir, que recebe uma string que corresponde a uma linha do arquivo, e retorna os dados armazenados em um objeto da estrutura **Aluno** descrita acima.

**turma.c**

```

1 #include "turma.h"
2 #include <stdlib.h>
3 Funcionario le_funcionario(char *linha) {
4     Aluno a; int i;
5     // Aloque, inicialmente, espaco para 5 caracteres no nome e matricula:
6     int n = 5;
7     a.matricula = _____;
8     a.nome = _____;
9     a.matricula[0] = '\0'; a.nome[0] = '\0';
10    for (i = 0; linha[i] != _____; i++) { // Ate onde vai a matricula?
11        // Aumente o espaco reservado em 5 posicoes (dica: 'n' nao muda):
12        a.matricula = _____;
13        a.matricula[i] = linha[i];
14        a.matricula[i + 1] = '\0';
15    }
16    // 'i' sai do loop anterior como sendo o indice do delimitador.
17    int j = 0;
18    for (i = i + 1; linha[i] != _____; i++, j++) { // Ate onde vai o nome?
19        // Aumente o espaco reservado em 5 posicoes (dica: 'n' nao muda):
20        a.nome = _____;
21        a.nome[j] = linha[i];
22        a.nome[j + 1] = '\0';
23    }
24    int mult = 1; a.nota = 0;
25    // 'i' sai do loop anterior como sendo o indice do delimitador.
26    for (int j = strlen(linha) - 1; j > i; j--) {
27        a.nota += (_____) * mult; // Converta os caracteres para inteiro
28        mult *= 10;
29    }
30    return a;
31 }

```

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

**turma.c**

```

31 Turma *le_turma(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 Turma.
35     Turma *t = _____;
36     t->alunos = NULL; t->n = 0; // Inicialmente nao temos nenhum aluno.
37     char *linha = le_linha(p);
38     while (linha != NULL) {
39         Aluno a = le_aluno(linha);
40         // Abra espaco no arranjo 'alunos' para UM novo aluno:
41         t->n = t->n + 1;
42         int bytes = t->n * _____;
43         t->alunos = _____;
44         t->alunos[t->n - 1] = a;
45         linha = le_linha(p);
46     }
47     _____; return t; // Feche o arquivo
48 }

```

5. (6 pontos) Crie uma função **RECURSIVA** chamada **indice\_k** que recebe um parâmetro do tipo **Turma** e dois inteiros *i* e *k*. Sua função deve retornar o índice do primeiro aluno a partir do índice *i* tal que todos os alunos deste em diante possuem nota  $\geq k$  (ou -1 caso não encontre nenhum aluno com nota  $\geq k$ ). Assuma que os alunos da turma estão ordenados de forma crescente. **Sua função NÃO pode conter laços** como **for**, **while** ou **do-while**.