Nome:..... Turma:.....

Omita os cabeçalhos de bibliotecas dos seus programas.
Estruture e comente o código, e não esqueça da identação.

Questão 1: Estruturação de programas

Uma clínica médica trabalha com uma agenda semanal de marcação de consultas. Cada paciente tem os seguintes dados pessoais armazenados: nome, endereço e convênio. Para cada consulta, as seguintes informações são usadas: nome do paciente (string), nome do médico (string), horário da consulta (estrutura). A clínica suporta o atendimento de no máximo 16 pessoas por dia (consultas de meia hora), trabalha durante 5 dias na semana.

a) **(1 ponto)** Defina as estruturas de dados adequadas para o problema.

b) **(2 pontos)** Estruture um programa de manipulação desta agenda, que possibilite a marcação de consultas e a modificação de médicos da consulta de um determinado horário. Ao final do programa, deve ser gerado um relatório contendo o número total de consultas do dia.

Questão 2: Passagem de parâmetros

Considere uma função **manipula_vetor** com três argumentos: um vetor de inteiros **v**, seu tamanho **n**, e um número inteiro **a.**

a) **(1 ponto)** Escreva essa função de modo que ela some cada um dos elementos do vetor ao número inteiro **a**, imprimindo-os na tela e escrevendo o resultado no próprio vetor **v**. Indique como seria uma chamada a essa função.

b) **(1 ponto)** Escreva essa função de modo que ela leia o valor da variável **a** do teclado dentro da função, some cada um dos elementos do vetor a esse valor, imprimindo-os na tela O vetor **v** NÃO deve ser modificado, e o valor de **a** DEVE SER modificado após a saída da função. Indique como seria uma chamada a essa função.

c) **(1 ponto)** Escreva uma versão **com retorno** da função b), de modo que ela retorne, no ponto da chamada, o vetor modificado. Indique como seria uma chamada a essa função.

Questão 3: Arquivos

Considere o arquivo texto profissões.txt mostrado abaixo. A primeira linha do arquivo contém o total de linhas que vêm a seguir. Cada linha possui três informações: o código de uma profissão (valor inteiro), o nome da profissão (string) e a taxa de desconto dadas em % (float).

```
4
100 alfaiate 2.4
101 analista 10.8
200 antropologo 1.6
1600 domestico 8
```

VIRE A PÁGINA →

Considere ainda a seguinte estrutura de dados e declaração de variáveis:

```
#define MAX_CONTRIBUINTES 30
struct TRABALHADOR{
  int codigo;
  char profissao[30];
  float TaxaDesconto;
};
int main()
{
   struct TRABALHADOR contribuintes[MAX_CONTRIBUINTES];
   ...
```

a) **(1 ponto)** Escreva um procedimento que receba um string com o nome do arquivo, abra e leia o arquivo acima e armazene seu conteúdo na estrutura de dados definida.

b) **(1 ponto)** Escreva um procedimento que crie um arquivo binário com nome "profissoes_selecionadas.dat" e armazene nesse arquivo os registros correspondentes apenas às profissões que têm taxas de desconto menores que 5%.

c) **(1 ponto)** Escreva um procedimento que crie um arquivo texto chamado "profissões_E.txt" e armazene nesse arquivo apenas os campos profissao e TaxaDesconto correspondentes às profissões que começam com a letra 'E'.

d) Escreva um procedimento que abra e leia o arquivo binário criado no item b), conte o número de linhas deste arquivo e imprima na tela este valor.

Questão 4: Recursividade

(1 ponto) Considere uma função recursiva **int func_soma(int n)** que recebe como entrada um número **n** par ou impar. Se **n** for par, a função deve retornar o valor 0 + 2 + 4 + ... + **n**, e se **n** for ímpar, a função deve retornar 1 + 3 + 5 + ... + **n**.

Funções auxiliares

strcmp(s1, s2) // retorna 0 se os strings forem iguais.

strcpy(string_destino, string_origem); // copia string origem no string destino

fread (<&varbuffer>,<numbytes>,<quant>,<FILE *>) // arquivo binario

fwrite (<&varbuffer>,<numbytes>,<quant>,<FILE *>) // arquivo binario

fgets (<&strbuffer>,<tamanho>,<FILE*>) // arquivo texto

fputs (<&strbuffer>,<FILE*>) // arquivo texto

fprintf (<FILE *>,<"txtE%cods">,<var1,var2,...>) // arquivo texto

fscanf (<FILE *>,<"%cods">,<&var1,&var2,...>) // arquivo texto

rewind(<FILE *>)

fseek(<FILE *>,<numbytesdesloc>,<SEEK_SET,SEEK_CUR,SEEK_END>)