| INF 1007 – P2 – 23/10/10 | |
|---|---|
| Nome: | |
| Matrícula: | Turma: |

## Instruções:

1. Esta prova deverá ser resolvida em até 90 minutos (1 hora e 30 minutos). Todas as questões deverão ter suas soluções implementadas em C, compiladas e testadas. Ao final da prova, os arquivos contendo o código fonte de cada solução serão recolhidos para a correção.

2. O aluno deve copiar o arquivo **TURMA_MATRICULA_P2.c** (onde deverá ser implementada a solução da prova) para seu *desktop* e RENOMEÁ-LO (dentro do visualC), substituindo TURMA pela turma do aluno (com 3 dígitos) e MATRICULA pela matrícula do aluno (com 7 dígitos). Por exemplo, "33B_0720870_P2.c" seria o arquivo contendo a solução da prova de um aluno da turma 33B, cuja matrícula é 0720870.

3. O arquivo deve conter o seguinte cabeçalho, onde WWWW deve ser substituído pelo nome do aluno, XXXXXXX pela matrícula do aluno (com 7 dígitos), YYY pela turma do aluno (com 3 dígitos):

```
/*-------------------------------------------------------------*/
/* Aluno: WWWWW                                                */
/* Matricula: XXXXXXX                                          */
/* Turma: YYY                                                  */
/*-------------------------------------------------------------*/
```

4. Durante o tempo de prova o aluno deve testar as funções que implementar. Para ajudá-lo nos testes, o aluno pode copiar para seu desktop (e utilizar) os seguintes arquivos:

    1) **Teste.c** - programa que usa as funções a serem implementadas como solução da prova
    2) **Compra.h** - interface do TAD *Compra*
    3) **Compra.c** - implementação do TAD *Compra*
    4) **Data.h** – definição do tipo estruturado *Data* usado na solução da prova
    5) **Prova.h** - cabeçalho das funções a serem implementadas pelo aluno

    Entretanto, além do texto relativo ao cabeçalho de identificação apresentado no item 3 (acima), o arquivo com a solução da prova deve conter APENAS o código fonte das funções solicitadas e implementadas. Ou seja, nesse arquivo NÃO pode ser incluído o código utilizado para testes.

5. Ao final dos 90 minutos de prova, serão concedidos mais 30 minutos para que cada aluno transcreva as soluções implementadas por ele para o corpo da prova, usando caneta preta ou azul. Estas soluções transcritas para o papel pelo aluno deverão ser entregues aos fiscais antes do aluno deixar a sala e têm a finalidade de prover um "back up" da solução em arquivo. A versão em papel SÓ será corrigida se ocorrer algum problema com o arquivo fornecido pelo aluno. O aluno deverá colocar seus dados de identificação na folha da prova.

6. Cada aluno é responsável por conferir o conteúdo do arquivo gravado pelo fiscal da prova. As provas cujo arquivo fornecido pelo aluno estiver em branco, mesmo que haja uma cópia escrita, NÃO serão consideradas.

*Não separe as folhas deste caderno.*

| INF 1007 – P2 – 23/10/10 | |
|---|---|
| Nome: | |
| Matrícula: | Turma: |

Considere a existência de um cadastro composto pelas compras realizadas em uma livraria on-line no último ano. Esse cadastro é representado por um vetor de PONTEIROS para o TAD *Compra*.

A interface do TAD *Compra* está definida no arquivo *Compra.h* onde se destacam as seguintes funções:

| Função | Objetivo |
|---|---|
| `char *cmp_obtem_cpf(Compra *c);` | Retorna o cpf do cliente que realizou determinada compra. O ponteiro da compra é recebido como parâmetro. O cpf é representado como uma cadeia de caracteres. |
| `Data cmp_obtem_data(Compra *c);` | Retorna a data de determinada compra. O ponteiro da compra é recebido como parâmetro. A data da compra é do tipo estruturado Data. |
| `char *cmp_obtem_cliente (Compra *c);` | Retorna o nome do cliente que realizou determinada compra. O ponteiro da compra é recebido como parâmetro. O nome do cliente é representado como uma cadeia de caracteres. |
| `float cmp_obtem_valor (Compra *c);` | Retorna o valor de determinada compra. O ponteiro da compra é recebido como parâmetro. |

Usando funções do TAD *Compra* implemente as funções apresentadas nas questões a seguir.

Protótipos de funções que podem ser úteis:

**stdio.h:**
```
int scanf (char* formato, ...);
int printf (char* formato, ...);
FILE* fopen (char* nome, char* modo);
int fclose (FILE* fp);
int fscanf (FILE* fp, char* formato, ...);
int fprintf (FILE* fp, char* formato, ...);
char* fgets(char* str, int size, FILE* fp));
int sscanf(char* str, char* formato, ...);
```

**stdlib.h:**
```
void* malloc (int nbytes);
void* realloc (void *p, int nbytes);
void  free (void* p);
```

**math.h:**
```
double sqrt (double x);
double pow (double x, double exp);
double cos (double radianos);
double sin (double radianos);
```

**string.h:**
```
int strlen (char* s);
int strcmp (char* s, char *t);
char* strcpy (char* destino, char* fonte);
char* strcat (char* destino, char* fonte);
```

*Não separe as folhas deste caderno.*

| INF 1007 – P2 – 23/10/10 | |
|---|---|
| Nome: | |
| Matrícula: | Turma: |

## Questão 1 – 1,5 pontos

Implemente em C a função *compara_datas* que:

- recebe duas variáveis (*d1* e *d2*) do tipo estruturado *Data*;
- retorna:
  - → -1, se a primeira data antecede cronologicamente a segunda data;
  - → 1, se a primeira data vem depois da segunda data;
  - → 0, se as duas datas são iguais.

O protótipo dessa função é o seguinte:

```
int compara_datas(Data d1, Data d2);
```

onde *Data* corresponde ao tipo estruturado descrito abaixo.

```
struct data {
      int dia, mes;
} ;
typedef struct data Data;
```

## Questão 2 – 3,5 pontos

Implemente em C a função *ordena* que:

- receba um vetor de PONTEIROS para o TAD *Compra* (*vpc*) e o número de elementos desse vetor (*n*).
  - → O vetor de compras (*vpc*) pode ter várias compras de um mesmo cliente, correspondentes a diferentes compras efetuadas por ele durante o ano. Um cliente pode realizar várias compras em uma mesma data. O vetor de compras (*vpc*) a princípio NÃO está ordenado.
- coloque o vetor em ordem crescente de cpf do cliente e, para o mesmo cpf, em ordem decrescente de data de compra, ou seja, as compras mais recentes do cliente devem aparecer primeiro. O cpf do cliente é representado como uma cadeia de caracteres e a data de compra é do tipo estruturado *Data*.
  - → A função deve implementar um dos métodos de ordenação apresentados no curso, NÃO podendo utilizar a função *qsort* da biblioteca padrão do C.
  - → A função *ordena*, usando ou não funções auxiliares, deve OBRIGATORIAMENTE chamar a função *compara_datas* da questão anterior.
  - → A função *ordena* deve OBRIGATORIAMENTE chamar as devidas funções do TAD *Compra*.

O protótipo dessa função é o seguinte:

```
void ordena(Compra **vpc, int n);
```

*Não separe as folhas deste caderno.*

| INF 1007 – P2 – 23/10/10 | |
|---|---|
| Nome: | |
| Matrícula: | Turma: |

## Questão 3 – 5,0 pontos

Implemente em C a função *busca_cliente* que:

- recebe o vetor de ponteiros para TAD *Compra* (*vpc*), o número de elementos desse vetor (*n*) e o cpf de um cliente (*cpf*), sendo que *cpf* é uma cadeia de caracteres.
  - → Considere que o vetor *vpc* está em ordem crescente de *cpf* e, para o mesmo cpf, em ordem decrescente de data de compra, conforme resultado da *questão 2*.
- busca no vetor (*vpc*) o cliente que possui o *cpf* recebido como parâmetro, utilizando o método de busca binária.
  - → NÃO pode ser usada a função *bsearch* da biblioteca padrão do C.
- caso o cliente seja encontrado, lista na tela todas as compras efetuadas por ele, em ordem decrescente de data de compra, seguindo o exemplo apresentado a seguir.
- retorna 1, se o cliente for encontrado, ou 0, caso contrário.
- A função *busca_cliente* deve OBRIGATORIAMENTE chamar as devidas funções do TAD *Compra*.

O protótipo dessa função é o seguinte:

```
int busca_cliente (Compra **vpc, int n, char *cpf);
```

Veja a seguir um exemplo do resultado da busca de cliente:

Conteúdo do vetor de compras, com 7 elementos:

"11111111111" - "bia nunes" - 40.40 - 19/5
"22222222222" - "leo silva" - 20.30 - 12/7
"22222222222" - "leo silva" - 60.30 - 20/5
"22222222222" - "leo silva" - 90.30 - 12/5
"77777777777" - "rui souza" - 80.50 - 20/6
**"99999999999" - "ana prado" - 50.00 - 20/5**
**"99999999999" - "ana prado" - 30.10 - 19/5**

Resultado apresentado na tela para o cpf 99999999999:

```
99999999999 - ana prado
20-5: 50.000000
19-5: 30.100000
```

*Não separe as folhas deste caderno.*