INF 1007 – P2D - 31/10/09	Questão 1
Nome:	Estação:
Matrícula:	Turma:

Uma transportadora mantém um cadastro com as principais informações sobre os produtos a serem entregues. Esses dados são representados como um vetor de ponteiros para o tipo *Produto*, definido a seguir:

```
struct produto {
   int cod;           /*  codigo do produto                       */
   char descr[31];    /*  descricao do produto                    */
   Data *entrega;     /*  data limite para a entrega do produto   */
   double valor;      /*  valor do produto                        */
   double multa;      /*  multa percentual por dias de atraso     */
};
typedef struct produto Produto;
```

O campo *entrega*, que representa o dia, mês e ano correspondentes à data limite para entrega do produto, é um ponteiro para o tipo estruturado *Data*, descrito a seguir:

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

a) [Valor: 2,0 pontos] Escreva uma função que calcula a multa diária devida pelo atraso na entrega de um produto. A função recebe como parâmetros a variável *prod*, do tipo *Produto*, e retorna o valor da multa diária, que é calculado como:

multa diária = valor x multa / 100

O protótipo da função é:

```
double Multa_diaria(Produto prod);
```

b) [Valor: 3,0 pontos] Usando uma das técnicas de ordenação apresentadas no curso, escreva uma função que recebe como parâmetros um vetor de ponteiros para *Produto*, através do ponteiro *vet* para seu primeiro elemento e um inteiro *n*, indicando o número de elementos, e coloque o vetor na ordem de prioridade de entregas, que corresponde à ordem cronológica de datas de entrega, com desempate pelo maior valor de multa diária (calculada como descrito no item anterior). Se ainda houver empate, o produto com o menor código vem na frente. O protótipo da função é:

```
void Ordena_produtos(Produto** vet, int n);
```

Não separe as folhas deste caderno. Todas as folhas devem ter seu nome. Responda cada questão na folha correspondente. Use o verso se necessário.

INF 1007 – P2D - 31/10/09	Questão 2
Nome:	Estação:
Matrícula:	Turma:

Um serviço de pesquisa genealógica determinou todos os membros masculinos de uma família até a última geração conhecida, representando as informações levantadas em um vetor de ponteiros para dados estruturados do tipo *Familiar*, descrito a seguir:

```
struct familiar {
   char nome[51];    /*  nome do familiar                */
   char pai[51];     /*  pai ou mae do familiar          */
};
typedef struct familiar Familiar;
```

a) [Valor: 2,0 pontos] Escreva uma função em C que recebe como parâmetros o ponteiro *vet*, para o primeiro elemento um vetor de ponteiros para *Familiar*, o inteiro *n*, indicando número de elementos desse vetor, e a cadeia de caracteres *nome*, representando o nome de um familiar, e retorna o índice no vetor do familiar que tem esse nome, se existir, ou -1, caso contrário. Considere que não há nomes repetidos. A função tem o seguinte protótipo:

```
int Indice_familiar(Familiar** vet, int n, char* nome);
```

b) [Valor: 3,0 pontos] Escreva uma função em C que recebe como parâmetros o ponteiro *vet*, para um vetor de ponteiros para *Familiar*, o inteiro *n*, indicando o número de elementos desse vetor, e duas cadeias de caracteres *nome1* e *nome2*, indicando dois familiares. A função deve retornar o grau de descendência entre o familiar *nome1* e o familiar *nome2*. Ou seja, se o campo *pai* do familiar *nome1* for igual a *nome2*, a função deve retornar 1. Se o *pai* do familiar *nome1* for descendente direto do familiar *nome2*, a função deve retornar 1 mais o grau de descendência entre *nome1* e *nome2*. Mas se *nome1* não for descendente direto de *nome2*, a função deve retornar 0. O patriarca da família não tem pai conhecido, ou seja, tem uma string vazia (“”) no campo *pai*. Por exemplo, se o vetor aponta para os familiares “Pedro” (*pai* = “Joao”), “Joaquim” (*pai* = “Manoel”), “Manoel” (*pai* = “Joao”), “Joao” (*pai* = “Antonio”) e “Antonio” (*pai* = “”), o grau de descendência entre Pedro e Antonio é 2, entre Joaquim e Antonio é 3, mas entre Joaquim e Pedro é 0. Considere que os dados fornecidos como parâmetros são sempre válidos, ou seja, *nome1* sempre existe no vetor, o vetor contém todos os elementos referenciados e *n* apresenta o valor correto. A função deve ter o seguinte protótipo:

```
int Descendencia(Familiar** vet, int n, char* nome1,
                 char* nome2);
```

Obs.: A função *Indice_familiar*, implementada no item *a*, pode ser utilizada pela função *Descendencia*.

Não separe as folhas deste caderno. Todas as folhas devem ter seu nome. Responda cada questão na folha correspondente. Use o verso se necessário.

RASCUNHO

Respostas nesta folha não serão consideradas.

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, ...);
```

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);
```

stdlib.h:

```
void* malloc (int nbytes);
void free (void* p);
void qsort (void *vet, int n, int tam, int (*comp) (const void*, const void*));
```

Não separe as folhas deste caderno.