INF 1007 – P2C - 31/10/09	Questão 1
Nome:	Estação:
Matrícula:	Turma:

Considere um programa que armazena os dados referentes aos candidatos de um concurso como um vetor de ponteiros para dados do tipo *Candidato*, definido conforme o tipo estruturado a seguir:

```
struct candidato {
   int num;            /*  numero de inscricao do candidato       */
   char nome[51];      /*  nome do candidato                      */
   Data *nasc;         /*  data de nascimento do candidato        */
   double geral;       /*  nota na prova de conhecimentos gerais  */
   double especifica;  /*  nota na prova especifica               */
};
typedef struct candidato Candidato;
```

O campo *nasc*, que representa o dia, mês e ano de nascimento do candidato, 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 média de um candidato. A função recebe como parâmetros a variável *cand*, do tipo *Candidato*, e retorna a média deste candidato, que corresponde à média ponderada da nota na prova geral, com peso 1, e a nota na prova específica, com peso 2. O protótipo da função é:

```
double Media_candidato(Candidato cand);
```

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 *Candidato*, através do ponteiro *vet*, para seu primeiro elemento, e um inteiro *n*, indicando seu número de elementos, e coloque o vetor na ordem de classificação no concurso, ou seja, na ordem decrescente de médias (calculadas como descrito no item anterior), com desempate pela maior nota na prova específica. Se as duas notas forem iguais, o candidato mais velho tem precedência. Se ainda houver empate, o candidato com o menor número de inscrição vem na frente. O protótipo da função é:

```
void Ordena_candidatos(Candidato** 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 – P2C - 31/10/09	Questão 2
Nome:	Estação:
Matrícula:	Turma:

Uma empresa de pesquisa genealógica determinou todos os membros consaguíneos 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 sexo;        /*  'M' para masculino e 'F' para feminino    */
   char progenitor[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 1 se o familiar *nome1* for descendente em linhagem masculina do familiar *nome2*, isto é, se o campo *progenitor* do familiar *nome1* for igual a *nome2* e ambos forem do sexo masculino, ou se o familiar *nome1* for do sexo masculino e seu progenitor for descendente em linhagem masculina do familiar *nome2*. A função deve retornar 0, caso contrário. O patriarca (ou matriarca) da família não tem progenitor conhecido, ou seja, tem uma string vazia (“”) no campo *progenitor*. Por exemplo, se o vetor aponta para os familiares “Pedro” (sexo = 'M', *progenitor* = “Antonio”), “Joaquim” (sexo = 'M', *progenitor* = “Maria”), “Maria” (sexo = 'F', *progenitor* = “Antonio”), “Antonio” (sexo = 'M', *progenitor* = “Severiano”) e “Severiano” (sexo = 'M', *progenitor* = “”), Pedro é descendente em linhagem masculina de Severiano, mas Joaquim não o é. Considere que os dados fornecidos como parâmetros são sempre válidos, ou seja, *nome1* sempre existe no vetor, o vetor contém todos os progenitores referenciados e *n* apresenta o valor correto. A função deve ter o seguinte protótipo:

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

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

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.