INF 1620 – P3 - 25/11/05	Questão 1
Nome:	
Matrícula:	Turma

Dizemos que uma matriz quadrada é um *quadrado mágico* se a soma dos elementos de cada linha, a soma dos elementos de cada coluna e a soma dos elementos de sua diagonal principal e secundária são todas iguais. Por exemplo, a seguinte matriz é um quadrado mágico:

$$\begin{bmatrix} 8 & 0 & 7 \\ 4 & 5 & 6 \\ 3 & 10 & 2 \end{bmatrix}$$

Escreva uma função que receba uma matriz quadrada como parâmetro (representada por um vetor simples) e verifique se a matriz é um quadrado mágico, retornando 1 caso seja uma quadrado mágico e zero no caso contrário. Essa função deve ter o seguinte protótipo:

```
int quadrado_magico (int n, int* matriz);
```

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 1620 – P3 - 25/11/05	Questão 2
Nome:	
Matrícula:	Turma

Considere a implementação de uma lista encadeada para armazenar números inteiros dada pelo tipo abaixo:

```
struct lista {
   int info;
   struct lista* prox;
};
typedef struct lista Lista;
```

Implemente uma função que receba duas listas e teste se a segunda lista é uma sub-lista da primeira, isto é, se a seqüência de valores armazenados na segunda lista aparece exatamente da mesma forma em algum ponto da primeira lista. A função deve retornar 1 para o caso da segunda lista ser uma sub-lista da primeira e zero no caso contrário, e deve ter o seguinte protótipo:

```
int sublista (Lista* l1, Lista* l2);
```

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 1620 – P3 - 25/11/05	Questão 3
Nome:	
Matrícula:	Turma

Considere uma lista duplamente encadeada para armazenar números reais, dada pelo tipo abaixo:

```
struct lista2 {
   float info;
   struct lista2* ant;
   struct lista2* prox;
};
typedef struct lista2 Lista2;
```

Implemente uma função que receba como parâmetros uma lista com seus elementos ordenados em ordem crescente e um número real *x*, e insira um novo nó na lista com o valor *x*, preservando a ordenação da lista. Essa função deve obedecer o seguinte protótipo:

```
Lista2* insere_ordenado (Lista2* l, float x);
```

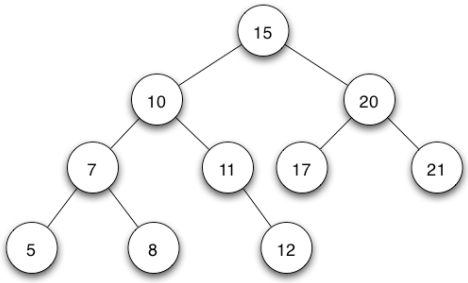
Não separe as folhas deste caderno. Todas as folhas devem ter seu nome. Responda cada questão na folha correspondente. Use o verso se necessário.

INF 1620 – P3 - 25/11/05	Questão 4
Nome:	
Matrícula:	Turma

Considere uma *árvore binária de busca* que armazena valores inteiros, onde os valores associados aos nós da sub-árvore à esquerda são menores que o valor associado à raiz e que os valores dos nós da sub-árvore à direita são maiores (considere que a árvore não possui valores repetidos). O tipo que representa um nó da arvore é dado por:

```
struct arv {
   int val;
   struct arv* esq;
   struct arv* dir;
};
typedef struct arv Arv;
```

Escreva uma função que receba uma árvore e um inteiro *x* e retorne um ponteiro para o nó da árvore que armazena o maior valor menor do que *x*. Caso *x* não seja encontrado na árvore ou se ele for o menor valor armazenado na árvore, a função deve retornar NULL. Por exemplo, se essa função fosse aplicada à seguinte árvore para *x* igual a 10, a função deveria retornar um ponteiro para o nó que armazena o valor 8:



A função deve tirar proveito da ordenação da árvore e obedecer ao seguinte protótipo:

```
Arv* antecessor (Arv* a, int x);
```

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);
```

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.