INF 1620 - P3 - 29/11/08	Questão 1
Nome:	
Matrícula:	Turma

[Valor: 3,0 pontos] Um dicionário eletrônico mantém seus dados em um vetor de ponteiros para o tipo *Verbete*, descrito a seguir:

```
struct verbete
{
   char palavra[51];
   char descricao[121];
};
typedef struct verbete Verbete;
```

Onde a cadeia de caracteres *palavra* é a palavra apresentada no verbete e a cadeia de caracteres *descricao* contém a definição dessa palavra. Considere que este vetor está ordenado em ordem alfabética com relação ao campo *palavra* de cada elemento.

Escreva uma função em C que efetue uma busca binária nesse vetor para localizar uma palavra. Essa função deve receber como parâmetros o vetor de verbetes (através do ponteiro *dic* para o primeiro elemento e um inteiro *n* indicando o número de elementos) e a cadeia de caracteres *pal*, que é a palavra que se deseja localizar, e deve escrever na tela "A palavra PPP significa DDD", onde PPP é a palavra encontrada e DDD a descrição correspondente. Se a palavra não for encontrada, a função deve escrever "Voce quis dizer XXX ou YYY?", onde XXX é a palavra do vetor que vem imediatamente antes da palavra procurada na ordem alfabética, enquanto YYY é aquela que vem imediatamente depois. Se a posição que a palavra procurada (e não encontrada) ocuparia no vetor for anterior à palavra do primeiro verbete, ou posterior à palavra do último verbete, a função deve imprimir "Voce quis dizer ZZZ?", sendo ZZZ a palavra do primeiro ou do último verbete do vetor, respectivamente. O protótipo da função é:

```
void Consulta(Verbete** dic, int n, char* pal);
```

Obs.: Sua função NÃO deve usar a função *bsearch* da biblioteca padrão:

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 - 29/11/08	Questão 2
Nome:	
Matrícula:	Turma

O sistema de controle de fila de um banco opera da seguinte forma. O cliente que chega à agência retira uma senha que pode ter prioridade preferencial (para gestantes, portadores de necessidades especiais e maiores de 65 anos) ou não preferencial. A agência dispõe de um caixa preferencial e diversos caixas não preferenciais. Ao decidir qual o próximo cliente a ser atendido, são consideradas duas políticas:

i. O caixa preferencial seleciona o primeiro cliente preferencial da fila. Não havendo clientes preferenciais, é selecionado o primeiro cliente da fila;

ii. O caixa não preferencial respeita a ordem de chegada e seleciona sempre o primeiro cliente que está aguardando na fila, preferencial ou não.

Para armazenar as informações da fila, o sistema utiliza uma lista duplamente encadeada, em que cada nó é do tipo *Cliente*, definido a partir da estrutura *cliente*, descrita a seguir:

```
struct cliente
{
   int senha;
   char prio;
   struct cliente *ant, *prox;
};
typedef struct cliente Cliente;
```

Nessa estrutura, o campo *senha* guarda o número da senha atribuída a um cliente, o campo *prio* é um caractere que pode ter os valores 'P' ou 'N', indicando que o cliente é preferencial ou não preferencial, e os campos *ant* e *prox* apontam para os nós anterior e próximo da lista, respectivamente.

a) [Valor: 2,0 ponto] Escreva uma função em C que receba como parâmetros o ponteiro *lst* para a lista de clientes, o inteiro *senha* e o caractere *prio*. Os dois últimos valores são usados para preencher os campos *senha* e *prio* de uma variável *Cliente* que deve ser alocada dinamicamente e inserida no final da lista. A função, que deve retornar o ponteiro para o início da lista atualizado, tem o seguinte protótipo:

```
Cliente* Enfileira(Cliente* lst, int senha, char prio);
```

b) [Valor: 2,0 pontos] Escreva uma função em C que receba como parâmetros o ponteiro *lst* para a lista de clientes e o caractere *prt* --- que tem o valor 'P' para indicar que a seleção deve ser feita para um caixa preferencial, ou 'N', para um caixa não preferencial --- e retorne a senha do cliente selecionado para atendimento, segundo as políticas apresentadas. Ou seja, se *prt* = 'P' a função deve retornar o primeiro cliente com campo *prio* = 'P' encontrado, ou, se não houver nenhum, o primeiro cliente da lista. Se *prt* = 'N' a função deve retornar o primeiro cliente da lista. O respectivo nó deve ser retirado da lista e liberado. Se a lista estiver vazia a função deve retornar -1. A função tem o seguinte protótipo:

```
int Seleciona(Cliente** lst, char prt);
```

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 - 29/11/08	Questão 3
Nome:	
Matrícula:	Turma

[Valor: 3,0 pontos] Um sistema de navegação armazena o caminho entre dois pontos como uma lista encadeada de dados do tipo *Logradouro*, descrito a seguir, onde o campo *nome* contém o nome do logradouro e o campo *prox* aponta para o próximo elemento da lista.

```
struct logradouro
{
   char nome[51];
   struct logradouro *prox;
};
typedef struct logradouro Logradouro;
```

Considere o tipo abstrato de dados *Pilha*, definido para armazenar ponteiros para cadeias de caracteres e que implementa as funções descritas na tabela a seguir:

Pilha* pilha_cria (void);	Retorna o ponteiro para uma nova pilha alocada dinamicamente.
char* pilha_pop (Pilha* p);	Retira um elemento do topo de uma pilha. O ponteiro da pilha é passado como parâmetro, e o retorno é o valor deste elemento.
void pilha_push (Pilha* p, char* x);	Insere um elemento no topo de uma pilha. O ponteiro da pilha e o elemento a ser inserido são passados como parâmetros.
int pilha_vazia (Pilha* p);	Verifica se a pilha está vazia. O ponteiro da pilha é passado como parâmetro e o retorno é 1, se a pilha está vazia, ou 0, caso contrário.
void pilha_libera (Pilha* p);	Esvazia e libera a memória alocada para uma pilha. O ponteiro da pilha é passado como parâmetro

Crie uma função em C para comparar dois caminhos e verificar se um é o inverso do outro, ou seja, comparar as duas listas que armazenam caminhos e verificar se o conteúdo de uma lista --- seqüência de nomes de logradouros armazenados em cada nó --- equivale ao da outra lista na ordem inversa. Essa função deve usar o tipo abstrato *Pilha* para auxiliar nessa comparação. Os parâmetros da função são os ponteiros *l1* e *l2* para as duas listas. O retorno da função deve ser 1 se uma lista for o inverso da outra e 0, caso contrário. A função tem o seguinte protótipo:

```
int Compara_listas(Logradouro* l1, Logradouro* 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.

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.