2ª Prova de Algoritmos e Estruturas de Dados I - Curso: Engenharia de Computação

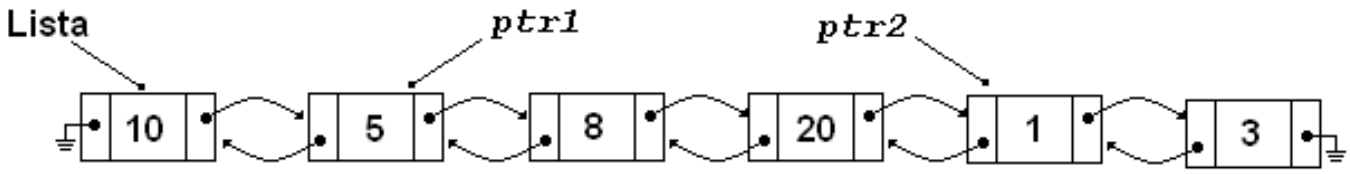
Nome do aluno: _____ **Data:** 14/08/2013

Resolva as questões abaixo identificando-as claramente na folha de respostas. Mantenha o silêncio na sala (mantendo desligado aparelhos eletrônicos). A interpretação das questões faz parte da prova. **Horário limite:** 15:40

(10 pontos) 1) O que é um algoritmo de ordenação estável? Considerando os algoritmos de ordenação estudados, diga quais são estáveis.

(20 pontos) 2) Seja uma lista duplamente encadeada, com a estrutura e exemplo abaixos:

```
typedef struct _nodo {
    struct _nodo *anterior;
    struct _nodo *proximo;
    int elemento;
} TipoNodo;
```



Sejam dados dois ponteiros para dois elementos distintos (ptr1 e ptr2). Assumindo que:

- Estes ponteiros existem e não serão passados valores nulos;
- Que podem estar em qualquer posição da lista;
- ptr1 vem "antes" na lista, de ptr2;
- Pode existir qualquer quantidade de elementos antes de ptr1, depois de ptr2 e entre ptr1 e ptr2.

Escreva uma função em C que recebe os ponteiros ptr1 e ptr2 e troque ptr1 por ptr2, sem copiar o conteúdo, ié, mexendo nos ponteiros.

(20 pontos) 3) Suponha dada uma lista encadeada que armazena números inteiros. Cada célula da lista tem a estrutura abaixo.

```
struct cel {
    int conteudo;
    struct cel *prox;
};
```

```
struct TipoLista{
    int qtd;
    struct cel *inicio;
};
```

Escreva uma função que transforme a lista em duas: a primeira contendo as células cujo conteúdo é par e a segunda aquelas cujo conteúdo é ímpar.

ATENÇÃO: A sua função NÃO deve alocar novos elementos, apenas manipular ponteiros.

(50 pontos) 4) Eu Posso Adivinhar a Estrutura de Dados!

Existe uma esrtutura de dados do tipo sacola, suportando duas operações:

- **1 x** - Jogue um elemento x na sacola.
- **2** - Tire um elemento da sacola.

Dada uma sequencia de operações que retornam valores, você vai adivinhar a estrutura de dados. É uma pilha, uma fila, uma fila de prioridade (sempre tire os elementos grandes por primeiro) ou qualquer outra coisa que você dificilmente consegue imaginar!

Entrada

Existem muitos casos de testes. Cada caso de teste começa com a linha contando um único inteiro n (1 <= n <= 1000). Cada uma das seguintes n linhas é um comando do tipo 1, ou um número inteiro 2, seguido de um número inteiro x. Isso significa que depois de executar um comando do tipo 2, obtemos um elemento x sem erros. O valor de x é sempre um número inteiro, positivo e não maior do que 100. O final da entrada é determinado pelo final do arquivo (EOF).

Saída

Para cada caso de teste, mostre um dos seguintes:

- **stack** - É definitivamente uma pilha.
- **queue** - É definitivamente uma fila.
- **priority queue** - É definitivamente uma fila de prioridade.
- **impossible** - Não pode ser uma pilha, uma fila ou uma fila de prioridade.
- **not sure** - Pode ser mais de uma das três estruturas mencionadas acima.

Exemplo de Entrada

```
6
1 1
1 2
1 3
2 1
2 2
2 3
6
1 1
1 2
1 3
2 3
2 2
2 1
2
1 1
2 2
4
1 2
1 1
2 1
2 2
7
1 2
1 5
1 1
1 3
2 5
1 4
2 4
```

Exemplo de Saída

```
queue
not sure
impossible
stack
priority queue
```