2ª Prova de Algoritmos e Estruturas de Dados I - Curso: Engenharia de Computação
**Data:** 18/10/2018 **Horário limite:** 15:30

**Perguntas comuns e suas respostas:**

- P: Tenho uma dúvida na questão tal.
  R: A compreensão do enunciado faz parte da prova.
- P: O que será corrigido?
  R: A lógica, a criatividade, a sintaxe, o uso correto dos comandos e dos tipos, os nomes das variáveis, a indentação, uso equilibrado de comentários no código e, evidentemente, a clareza. Nesta prova, você deverá sobretudo escrever códigos modulares, usando corretamente funções e/ou procedimentos, conforme o caso, além de uso correto de variáveis locais ou globais e a passagem de parâmetros por referência ou por valor.
- P: Posso fazer a lápis?
  R: Não. A prova deverá ser feita a caneta.
- P: Posso responder na folha de questões?
  R: Não. A prova deverá ser respondida na folha de respostas.

**(09 pontos)** 1) Diferencie Lista, Fila e Pilha.

**(18 pontos)** 2) Mostre a situação da fila F. Inicialmente vazia, após a execução de cada uma das operações a seguir:

```
1  Enfila(&F,'a');
2  Enfila(&F,'a');
3  Enfila(&F,'b');
4  Enfila(&F,'c');
5  Desenfila(&F);
6  Enfila(&F,'e');
7  Enfila(&F,Espia(&F));
8  Enfila(&F,Desenfila(&F));
9  Desenfila(&F);
10 Enfila(&F,Espia(&F));
11 Enfila(&F,Enfila(&F,'g')+'a');
12 Enfila(&F,Desenfila(&F)+Espia(&F));
```

- A função 'Espia' retorna o elemento que seria desenfilado, mas não desenfila o elemento;
- A função 'Desenfila' retorna o elemento desenfilado;
- A função 'Enfila' retorna 1 quando consegue enfilar e 0 quando não consegue.
  - Você pode assumir que a fila, neste exercício, possui tamanho suficiente para enfilar todos os elementos.

**(15 pontos)** 3) Sobre algoritmos de ordenação:

a) É possível afirmar que o algoritmo QuickSort é superior aos algoritmos elementares (ex: BubbleSort, InsertionSort, SelectionSort) em todos os casos? Porquê?

b) Explique alguma estratégia para minimizar o problema apresentado no item (a).

c) A respeito da estabilidade de um algoritmo de ordenação, defina o conceito de estabilidade e cite 1 algoritmo estável e 1 não estável.

**(40 pontos)** 4) Carteiro Um carteiro é o responsável por entregar as encomendas na rua de Joãozinho. Por política da empresa, as encomendas devem ser entregues na mesma ordem que foram enviadas, mesmo que essa não seja a forma mais rápida. Cansado de subir e descer aquela rua tantas vezes, nosso amigo quer mostrar à empresa quanto tempo ele leva para entregar as encomendas, na tentativa de derrubar essa política.

A rua de Joãozinho tem N casas. Naturalmente, as casas são numeradas de forma ordenada (não necessariamente por números consecutivos). Como as casas possuem aproximadamente o mesmo tamanho, você pode assumir que o carteiro leva uma unidade de tempo para caminhar de uma casa até a casa imediatamente vizinha.

Há M encomendas para essa rua, que devem ser entregues na mesma ordem em que chegaram. Cada encomenda contém o número da casa onde deve ser entregue.

Escreva um programa que determine quanto tempo o carteiro levará para entregar todas as enco- mendas, assumindo que quando o tempo começa a contar, ele está na primeira casa (a de menor número), e o tempo termina de contar quando todas as encomendas foram entregues (mesmo que o carteiro não esteja de volta na primeira casa). Você pode desprezar o tempo para colocar a enco- menda na caixa de correio (ou seja, se ele só tiver uma encomenda, para a primeira casa, a resposta para o problema é zero).

**Entrada**

A primeira linha contém dois inteiros, N e M , respectivamente o número de casas e o número de encomendas. A segunda linha contém N inteiros em ordem estritamente crescente, indicando os números das casas. A terceira linha contém M inteiros indicando os números das casas onde as encomendas devem ser entregues, na ordem dada na entrada.

**Saída**

Seu programa deve produzir uma única linha, contendo um único inteiro, o tempo que o carteiro levará para entregar todas as encomendas na ordem correta, assumindo que ele começa na casa de menor número.

Restrições

- $1 \leq N \leq 45.000$ e $1 \leq M \leq 45.000$
- O número de cada casa é um inteiro entre 1 e 1.000.000.000

Exemplos

```
Entrada
5 5
1 5 10 20 40
10 20 10 40 1
Saída
10
```

```
Entrada
3 4
50 80 100
80 80 100 50
Saída
4
```

**(20 pontos)** 5) Durante o desenvolvimento de um sistema uma fila foi utilizada para armazenar os números referentes ao processamento, porém quando o processamento é encerrado é necessário que seja impresso na tela todos os itens dessa fila ao contrário. Implemente uma função `void imprime-fila-ao-contrario()`, que imprime na tela o conteúdo da fila ao contrário da ordem armazenada. Você pode utilizar apenas as funções de Fila para manipular os elementos, ou seja, **NÃO** pode acessar os elementos da fila diretamente. Também a fila deve permanecer intacta após a execução da função `imprime-fila-ao-contrario()`.

Seguem os protótipos das funções que estão disponíveis e que podem ser utilizadas:

- `void enfilera(struct TipoFila *f)`
- `int desenfilera(struct TipoFila *f)`
- `int esta-vazia(struct TipoFila *f)`

Exemplo:

Imagine uma fila com os elementos armazenados:

```
10 20 -3 -15 20 50
```

Deve aparecer na tela:

```
50 20 -15 -3 20 10
```