Algoritmos e Programação I: Lista de Exercícios 07-2 - Orientações para submissão no Moodle e BOCA. *

Faculdade de Computação
Universidade Federal de Mato Grosso do Sul
79070-900 Campo Grande, MS
`http://moodle.facom.ufms.br`

Após submeter no BOCA, não se esqueça de enviar o programa para o Moodle, usando o seguinte nome: pxxloginname.c, onde xx é o número do problema e o loginname é o seu login.

Compile o seu programa usando:

```
gcc -Wall -pedantic -std=c99 -o programa programa.c [-lm]
```

A flag `-lm` deve ser usada quando o programa incluir a biblioteca `math.h`.

Uma forma eficiente de testar se o seu programa está correto é gerando arquivos de entrada e saída e verificar a diferença entre eles. Isso pode ser feito da seguinte forma:

```
./programa < programa.in > programa.out
```

O conjunto de entrada deve ser digitado e salvo num arquivo `programa.in`. O modelo da saída deve ser digitado e salvo num arquivo `programa.sol`.

Verifique se a saída do seu programa `programa.out` é EXATAMENTE igual ao modelo de saída `programa.sol`. Isso pode ser feito com a utilização do comando `diff`, que verifica a diferença entre dois arquivos.

```
diff programa.out programa.sol
```

O seu programa só está correto se o resultado de `diff` for vazio (e se você fez todos os passos como indicado).

A solução dos exercícios deve seguir a metodologia descrita em sala:

a. Diálogo

b. Saída/Entrada (pós e pré)

c. Subdivisão

d. Abstrações

*Este material é para o uso exclusivo da disciplina de Algoritmos e Programação I da FACOM/UFMS e utiliza as referências bibliográficas da disciplina (B. Forouzan e R. Gilbert, A. B. Tucker et al. e S. Leestma e L. Nyhoff, Lambert et al., H. Farrer et al., K. B. Bruce et al., e Material Didático do Prof. Fábio Henrique Viduani Martinez.

e. Implementação

f. Teste

Na submissão ao BOCA não se esqueça de comentar todos os `printf`'s da entrada e que a saída não pode conter acentuação ou *ç*.

Exercícios

1. Um implementação possível para uma pilha é através da utilização de um array `Stack` com `StackLimite` elementos e o topo da pilha na posição 1 do array.

   a. Porque esta não é uma boa implementação?

   b. Um implementação melhor é o de deixar a pilha crescer a partir da posição 1 até a posição `StackLimite` e manter uma variável `Top` para apontar para o topo corrente da pilha. Escreva funções para as operações de `push` (empilhar) e `pop` (desempilhar) nesta implementação.

   c. Use as funções da parte **b.** em um programa que leia um comando `I` (Inclui) ou `R` (Remove); para `I`, leia um inteiro e empilhe (`push`) o inteiro lido na pilha; para `R`, desempilhe (`pop`) o inteiro lido da pilha e imprima a pilha.

2. Para uma fila, podemos utilizar a idéia de implementação de uma pilha no exercício 8, usando o array Queue com QueueLimite elementos e mantendo dois apontadores `Início` para o item do início da fila e `Final` para o item no final. Para incluir um item na fila, simplesmente incremente `Final` por 1 e armazene o item em `Queue[Final]`; para remover um item, simplesmente incremente `Início` por 1.

   a. Descreva as inadequações desta implementação. Considere, por exemplo, um array de inteiros com 5 elementos e a seguinte seqüência de operações na fila: `Inclua 37`, `Inclua 82`, `Inclua 59`, `Remova` um item, `Remova` um item, `Inclua 66`, `Inclua 13`, `Inclua 48`.

   b. Uma implementação melhor é a de pensar o array como sendo circular, com o primeiro elemento seguindo o último. Para isto, utilize adição função `QueueLimite`. Escreva funções para as operações de `Inclusões` e `Remoções`, utilizando esta implementação. Use estas funções em um programa que leia um comando `I` (Inclui) ou `R` (Remove); para `I`, leia um inteiro e adicione o inteiro lido na fila; para `R`, remova um inteiro da fila e imprima a fila.

3. A Companhia de Erva Mate Pantaneira registra o número de pacotes de erva mate produzidos cada dia durante um período de 4 semanas. Escreva um programa com funções que leia estes números de produção e armazene-os em um array. O programa deve então aceitar como entrada do usuário um número para a semana e um número para o dia e então deve imprimir o nível de produção para aquele dia. Assuma que cada semana possua 5 dias de trabalho.

4. A Companhia de Erva Mate Pantaneira produz vários tipos de erva mate, cada um identificado por um número de produto. Escreva um programa com funções que leia em dois arrays, `Número` e `Preço`, onde `Número[0]` e `Preço[0]` são o número do produto e o preço unitário para o primeiro item, `Número[1]` e `Preço[1]` são o número do produto e o preço unitário para o segundo item, e assim sucessivamente. O programa deve proporcionar que o usuário selecione uma das seguintes opções:

a. Recupere e imprima o preço de um produto cujo número é fornecido pelo usuário. (Utilize a função de busca linear do Exercício 6 para determinar o índice no array Número do item especificado.)

b. Imprima uma tabela mostrando o número do produto e o preço de cada item.

5. A Companhia de Erva Mate Pantaneira mantém dois depósitos, um em Campo Grande e outro em Ponta Porã. Cada um com estoques de no máximo 25 itens diferentes. Escreva um programa com funções que leia os números dos produtos dos itens armazenados no depósito em Campo Grande e armazene-os no array CampoGrande, e então repita este procedimento para os itens armazenados no depósito de Ponta Porã, armazenando esses números de produtos no array PontaPorã. O programa deve então encontrar e imprimir a interseção destas duas listas de números, isto é, a coleção de números de produtos comuns a ambas as listas. As listas não devem ser assumidas como tendo o mesmo número de elementos.

6. Repita o exercício 5, mas encontre e imprima a união das duas listas, isto é, a coleção de números de produtos que são elementos de pelo menos uma das listas.

7. Se $\bar{x}$ denota a média dos números $x_1, x_2, \ldots, x_n$, a variância (σ^2) é a média dos quadrados dos desvios dos números da média:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

e o desvio padrão é a raiz quadrada da variância. Escreva um programa com funções para ler uma lista de números reais e contá-los, então calcular a sua média, variância e desvio padrão. Imprima com cabeçalhos e rótulos apropriados quantos números foram lidos, sua média, variância e desvio padrão.

8. Ordenação por inserção é um método eficiente de ordenação para pequenos conjuntos de dados. Ele inicia com o primeiro item x_0, então insere x_1 nesta lista de um item na posição correta de tal forma que a lista destes dois elementos forme uma lista ordenada, então insere x_2 nesta lista de dois elementos na posição correta, e assim por diante. Por exemplo, para ordenar a lista $\{7, 1, 5, 2, 3, 4, 6, 0\}$, os passos são os seguintes (o elemento sendo inserido está sublinhado):

   Lista

```
7
1, 7    (movimente 7 uma posição para a direita)
1, 5, 7   (movimente 7 para a direita novamente)
1, 2, 5, 7   (movimente 5 e 7 para a direita)
1, 2, 3, 5, 7   (movimente 5 e 7 para a direita)
1, 2, 3, 4, 5, 7    (movimente 5 e 7 para a direita)
1, 2, 3, 4, 5, 6, 7    (movimente 7 para a direita)
0, 1, 2, 3, 4, 5, 6, 7    (movimente todos de 1 até 7 para a direita)
```

   Escreva um programa usando uma função para ordenar uma lista de itens, utilizando o método de ordenação por inserção.

9. A companhia de investimento Caiman & Caiman vem registrando os preços de venda para um determinado produto agrícola por um período de 15 dias. Escreva um programa, usando funções, que leia estes preços e ordene-os em ordem crescente, utilizando o método de ordenação por inserção descrito no exercício anterior. O programa deve imprimir a variação do preço do produto agrícola, isto é, o maior e o menor preço registrado, e também o preço médio.

10. Ordenação por seleção não é um esquema eficiente para ordenar listas grandes. Uma das razões para sua ineficiência é que somente um item é movimentado de cada vez para sua correta posição em cada exploração na lista. `Shell sort` (Donald Shell) tenta melhorar isto através da movimentação de diversos itens para suas posições corretas. (Neste exercício apresentamos uma versão mais detalhada do método visto em sala). Em cada exploração da lista, itens que estão a uma distância especificada uns dos outros são comparados, e se eles não estão em ordem (um item maior precede um menor), eles são intercambiados. Quando não mais intercâmbios são efetuados para uma dada distância, a distância é dividida ao meio, e a exploração compare-intercambie continua. A distância inicial é usualmente tomada como $n/2$ para uma lista de n itens. Por exemplo, para a lista

$$6, 1, 5, 2, 3, 4, 0$$

a sequência seguinte de distâncias e explorações será usada:

# Exploração	Distância	Lista Rearranjada	Intercâmbios
1	3	2, 1, 4, 0, 3, 5, 6	(6, 2), (5, 4), (6, 0)
2	3	0, 1, 4, 2, 3, 5, 6	(2, 0)
3	3	0, 1, 4, 2, 3, 5, 6	Nenhuma
4	1	0, 1, 2, 3, 4, 5, 6	(4, 2), (4, 3)
5	1	0, 1, 2, 3, 4, 5, 6	Nenhuma

Escreva um programa para ordenar uma lista de itens, utilizando o método `Shell sort`.

11. Antônio, o carteiro ficou enfadonho uma certa noite, e para quebrar a monotonia da noite, ele executou a seguinte experiência com uma fila de caixas postais no correio. Estas caixas postais eram numeradas de 1 a 150, e começando na caixa postal 2, ele abriu as portas de todas as caixas postais com números pares. A seguir, começando na caixa postal 3, ele foi para toda terceira caixa postal, abrindo sua porta se estivesse fechada, fechando-a se estivesse aberta. Então ele repetiu este procedimento para todos as quartas caixas postais, então todas quintas caixas postais, e assim por diante. Quando ele terminou, ele ficou surpreso com a distribuição das caixas postais fechadas. Escreva um programa para determinar quais caixas postais ficaram fechadas.

12. Escreva um programa para adicionar dois inteiros grandes de comprimento até 300 dígitos. Uma abordagem é o de tratar cada número como uma lista, onde cada elemento é um bloco de dígitos do número. Por exemplo, o inteiro 179534672198 pode ser armazenado com `Bloco[0] = 198`, `Bloco[1] = 672`, `Bloco[2] = 534` e `Bloco[3] = 179`. Então adicione os inteiros elementos por elementos, levando o vai um para o próximo elemento quando for o caso.

13. Procedendo como no exercício 20, escreva um programa para multiplicar dois inteiros grandes de comprimento até 300 dígitos.

14. (cheia.c) [fhvm] Uma sequência de $n > 0$ números inteiros é chamada de **cheia** se os valores absolutos das diferenças entre os elementos consecutivos representam todos os possíveis valores entre 1 e $n - 1$.

    Exemplo:

    > 1 4 2 3 é uma sequência cheia, porque os valores absolutos das diferenças entre seus elementos consecutivos são 3, 2 e 1, respectivamente.

    Observe que esta definição implica que qualquer sequência contendo exatamente um número inteiro é uma sequência cheia.

    a. Escreva uma função com a seguinte interface:

    ```
    int uns(int n, int cont[MAX])
    ```

    que receba um número inteiro n e um vetor de números inteiros com n elementos, e devolva 1 se cada elemento do vetor é igual a 1. Caso contrário, a função devolve 0.

    b. Escreva um programa que receba um número inteiro $k > 0$ que representa a o número de casos de teste. Para cada caso de teste, receba um número inteiro $n > 0$ e uma sequência de n números inteiros, com $1 \leq n \leq 100$, e verifique se a sequência é cheia ou não. Em caso positivo, imprima `C` para essa entrada. Caso contrário, imprima `N`. Use a função do item (a).

    Exemplo de entrada:

    ```
    cheia.in
    2
    4 1 4 2 3
    5 1 4 2 -1 6
    ```

    Exemplo de saída:

    ```
    cheia.sol
    C
    N
    ```

15. (crivo.c) [fhvm] Um número primo é um inteiro maior que 1 cujos únicos divisores são 1 e o próprio inteiro. Um método para encontrar todos os números primos no intervalo de 1 até n é conhecido como Crivo de Eratóstenes. O **crivo de Eratóstenes**[1] é um método para encontrar números primos até um certo valor limite. Segundo a tradição, foi criado pelo matemático grego Eratóstenes, o terceiro bibliotecário-chefe da Biblioteca de Alexandria.

    Considere a lista dos números de 2 até n. Dois é o primeiro número primo, mas os múltiplos de 2 $(4, 6, 8, \ldots)$ não são, e então eles são marcados na lista. O primeiro número após o 2 que não foi marcado é 3, o próximo primo. Então marcamos na lista todos os múltiplos de 3 $(6, 9, 12, \ldots)$. O próximo número não marcado é 5, o próximo primo, e então marcamos todos múltiplos de 5 $(10, 15, 20, \ldots)$. Repetimos

[1] Eratóstenes de Cirene (276 a.C. − 195 a.C.), nascido em Cirene na Grécia, matemático, poeta, geógrafo, astrônomo e músico.

este procedimento até que o próximo número na lista que não tenha sido marcado tenha o seu quadrado maior que n. Todos os números na lista são os primos de 2 até n.

A ideia do método é começar com todos os inteiros no intervalo $[2, n]$ e eliminar, em cada iteração, os múltiplos próprios de um número primo considerado. O primeiro número primo a ser considerado é 2 e o último é $\lfloor\sqrt{n}\rfloor$. Dados dois números inteiros a e b, dizemos que a é **múltiplo próprio** de b se a é múltiplo de b e $a > b$.

Exemplo:

> Se $n = 25$, consideramos o vetor C com os $n - 1$ elementos a seguir
>
> $2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10,\ 11,\ 12,\ 13,\ 14,\ 15,\ 16,\ 17,\ 18,\ 19,\ 20,\ 21,\ 22,\ 23,\ 24,\ 25$.
>
> Eliminando os múltiplos próprios de 2 restam
>
> $$2,\ 3,\ 5,\ 7,\ 9,\ 11,\ 13,\ 15,\ 17,\ 19,\ 21,\ 23,\ 25\ .$$
>
> Eliminando os múltiplos próprios de 3 restam
>
> $$2,\ 3,\ 5,\ 7,\ 11,\ 13,\ 17,\ 19,\ 23,\ 25\ .$$
>
> Eliminando os múltiplos próprios de 5 restam
>
> $$2,\ 3,\ 5,\ 7,\ 11,\ 13,\ 17,\ 19,\ 23\ .$$
>
> Não há necessidade de verificar os múltiplos próprios de $7, 11, 13, 17, 19$ e 23, já que $\lfloor\sqrt{25}\rfloor = 5$.

a. Escreva uma função com a seguinte interface:

```
void elimina(int *m, int C[MAX], int i)
```

que receba um número inteiro $m > 0$, um vetor C de números inteiros com m elementos e um índice i, e elimine deste vetor os múltiplos próprios de $C[i]$ a partir da posição $i + 1$ de C. Devolva também o novo valor de m, a nova quantidade de elementos de C.

b. Escreva um programa que receba um número inteiro $k > 0$ que representa a quantidade de casos de teste. Para cada caso de teste, receba um número inteiro n, com $2 \leq n \leq 1000$, e imprima os números primos de 2 a n, usando a função do item (a).

Exemplo de entrada:

```
crivo.in
2
25
7
```

Exemplo de saída:

```
crivo.sol
2 3 5 7 11 13 17 19 23
2 3 5 7
```

Observação: na saída, na apresentação de uma sequência resultante de números primos, sempre informe um elemento e um espaço em uma linha; isso significa que o último elemento da linha vem sempre sucedido de um espaço e, então, de uma mudança de linha.

16. (intercala.c) [fhvm]

    a. Escreva uma função com a seguinte interface:

    ```
    void ordena_insercao(int m, int A[MAX])
    ```

    que receba um número inteiro m e um vetor A de números inteiros com m elementos, e coloque os elementos desse vetor em ordem crescente usando o método da inserção.

    b. Escreva uma função com a seguinte interface:

    ```
    void intercala(int m, int A[MAX], int n, int B[MAX], int* k,
                                                    int C[2*MAX])
    ```

    que receba um número inteiro $m > 0$, um vetor A de números inteiros com m elementos em ordem crescente, um número inteiro $n > 0$ e um vetor B de números inteiros com n elementos em ordem crescente, e compute um vetor C contendo os elementos de A e de B sem repetição e em ordem crescente; devolva no parâmetro k o total de elementos de C.

    c. Escreva um programa que receba um número inteiro $k > 0$, que representa a quantidade de casos de teste; para cada caso de teste, receba um número inteiro m, $1 \leq m \leq 100$, um conjunto X de m números inteiros distintos, um número inteiro n, $1 \leq n \leq 100$, e um conjunto Y de n números inteiros distintos; coloque os elementos dos conjuntos X e Y em ordem crescente usando a função do item (a) e intercale esses dois vetores resultantes, usando a função do item (b), obtendo como resultado um vetor de números inteiros em ordem crescente.

Exemplos de entrada:

```
intercala.in
3

8 11 3 9 1 15 13 7 5
8 4 14 16 8 2 12 10 6

6 2 4 3 1 6 5
6 12 9 8 11 7 10

5 2000 200000 20 200000 200
3 6 5 4
```

Exemplos de saída:

```
intercala.sol
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1 2 3 4 5 6 7 8 9 10 11 12
4 5 6 20 200 2000 200000 200000
```

Observação: na saída, na apresentação de um vetor resultante, sempre informe um elemento e um espaço; isso significa que o último elemento vem sempre sucedido de um espaço e, então, de uma mudança de linha.

17. (intersecao.c) [fhvm] Em um programa na linguagem C, um conjunto pode ser representado por um vetor da seguinte forma: `V[0]` contém o número de elementos do conjunto; `V[1]`, `V[2]`, ... são os elementos do conjunto, sem repetições.

    a. Escreva uma função com a seguinte interface:

    ```
    void intersec(int A[MAX+1], int B[MAX+1], int C[MAX+1])
    ```

    que dados dois conjuntos de números inteiros A e B, construa um terceiro conjunto C tal que $C = A \cap B$. Lembre-se de que em `C[0]` a sua função deve colocar o tamanho da interseção.

    b. Escreva um programa que receba um número inteiro $k > 0$, que representa a quantidade de casos de teste; para cada caso de teste, receba um número inteiro $n \geq 2$ e uma sequência de n conjuntos de números inteiros, com cada um com no máximo 100 elementos, e construa e imprima um vetor que representa a interseção dos n conjuntos. Se a interseção for vazia, imprima uma linha em branco.

    Observação: não leia todos os conjuntos de uma só vez. Leia os dois primeiros conjuntos e calcule a primeira interseção. Depois, leia o próximo conjunto e calcule uma nova interseção entre esse conjunto lido e o conjunto da interseção anterior, e assim por diante.

    Exemplo de entrada:

    ```
    intersecao.in
    2

    3
    5 7 8 2 1 9
    4 1 8 2 9
    3 7 1 9

    2
    2 1 5
    1 5
    ```

    Exemplo de saída:

    ```
    intersecao.sol
    1 9
    5
    ```

    Observação: na saída, na apresentação de um conjunto interseção, sempre informe um elemento e um espaço; isso significa que o último elemento vem sempre sucedido de um espaço e, então, de uma mudança de linha.

18. (prima.c) [fhvm] Um **número primo** é um número que possui somente dois divisores: ele mesmo e o número 1. Exemplos de números primos são: 2, 3, 5, 17, 101 e 10007.

    Considere uma palavra composta por uma sequência de caracteres alfabéticos minúsculos (`'a'`–`'z'`) e/ou maiúsculos (`'A'`–`'Z'`). Atribuímos valores numéricos a cada caractere de uma palavra da seguinte forma: o caractere `'a'` vale 1, o caractere `'b'` vale 2, e assim por diante, até o caractere `'z'`, que vale 26; o caractere `'A'` vale 27, o caractere `'B'` vale 28, e assim por diante, até o caractere `'Z'`, que vale 52. O **valor de uma palavra** é definido como a soma dos valores de cada um de seus caracteres, de acordo com a atribuição acima. Por fim, dizemos que uma palavra é uma **palavra prima** se seu valor é um número primo.

    a. Escreva uma função com a seguinte interface:

    ```
    int primo(int p)
    ```

    que receba um número inteiro p e devolva 1 se p é primo e 0 caso contrário.

    b. Escreva uma função com a seguinte interface:

    ```
    int valor(char palavra[MAX+1])
    ```

    que receba uma cadeia de caracteres e devolva o valor da palavra armazenada na cadeia.

    c. Escreva um programa que receba um número inteiro $k > 0$ que representa a quantidade de casos de teste. Para cada caso de teste, receba uma palavra contendo apenas caracteres alfabéticos minúsculos e/ou maiúsculos, com no máximo 20 caracteres, e, de acordo com a valoração de caracteres dada acima, verifica se a palavra é prima ou não. Em caso positivo, imprima `P` na saída. Caso contrário, imprima `N`.

    Exemplo de entrada:

    ```
    prima.in
    3
    UFMS
    prova
    analise
    ```

    Exemplo de saída:

    ```
    prima.sol
    P
    N
    P
    ```

19. (segmentos.c) [fhvm] Um **segmento** de uma sequência $x_0, x_1, \ldots, x_{k-1}$ de números inteiros é um trecho $x_j, x_{j+1}, \ldots, x_{j+n-1}$ da sequência tal que $0 \leq j, n < k$. Dois segmentos de uma sequência $x_0, x_1, \ldots, x_{k-1}$ são chamados de **segmentos consecutivos** se existem i e m tais que $x_i, x_{i+1}, \ldots, x_{i+m-1}$ e $x_{i+m}, x_{i+m+1}, \ldots, x_{i+2m-1}$ são segmentos da sequência. Dois segmentos consecutivos são ditos **iguais** se existem i e m tais que

    $$x_i, x_{i+1}, \ldots, x_{i+m-1} = x_{i+m}, x_{i+m+1}, \ldots, x_{i+2m-1} .$$

    Exemplo:

Na sequência $3, 7, 9, \underline{5, 4}, \underline{5, 4}, 8, 6$ temos $i = 3$ e $m = 2$ e, portanto, há dois segmentos consecutivos iguais $5, 4$ na sequência.

a. Escreva uma função com a seguinte interface:

```
int iguais(int k, int x[DIM], int i, int m)
```

que receba um número inteiro $k > 0$, um vetor x de números inteiros com k elementos e dois números inteiros i e m, com $0 \leq i, m < k$, e verifique se $x_i = x_{i+m}, x_{i+1} = x_{i+m+1}, \ldots, x_{i+m-1} = x_{i+2m-1}$, devolvendo 1 em caso positivo e 0 em caso negativo.

b. Escreva um programa que receba várias sequências de números inteiros tais que, em cada sequência, o primeiro de seus números indica a quantidade de elementos da sequência, e determine se a sequência tem dois segmentos consecutivos iguais. Em caso positivo, imprima o maior desses segmentos na saída. Empates são resolvidos arbitrariamente. Se a sequência não tem dois segmentos consecutivos iguais, imprima 0 (zero). A última sequência fornecida contém apenas o número 0 (zero).

Exemplo de entrada:

```
segmentos.in
8 7 9 5 4 5 4 8 6
3 1 5 2
0
```

Exemplo de saída:

```
segmentos.sol
5 4
0
```

Observação: na saída, na apresentação de um segmento resultante, sempre informe um elemento e um espaço em uma linha; isso significa que o último elemento da sequência vem sempre sucedido de um espaço e, então, de uma mudança de linha.