**MAC 115 – Introdução à Computação**

Instituto de Física – Segundo Semestre de 2000 - Diurno

2ª Prova – 8/12/2000

Nome do aluno: _____ Turma:_____

Assinatura: _____

Professor(a): _____

Nº USP: _____ Curso: _____

**Instruções:**

1. Não destaque as folhas deste caderno.
2. A prova pode ser feita a lápis. Cuidado com a legibilidade.
3. Há 3 questões na prova. Verifique antes de começar a prova se o seu caderno de questões está completo.
4. Não é permitido o uso de folhas avulsas para rascunho.
5. Nas questões que envolvem elaboração de programas, coloque comentários suficientes para que o programa seja facilmente compreendido.
6. Não é necessário apagar rascunhos no caderno de questões.

Não escreva nesta parte da folha

| Questão | Nota |
|---|---|
| 1 | |
| 2 | |
| 3 | |
| Total | |

**BOA SORTE!**

**Questão 1** (valor: 3.0)

Simule a execução do programa abaixo *destacando a sua saída* (o que vai sair na tela). Dados de entrada (a serem lidos):

```
7
3  6  1  4  7  8  5
```

```
/************************************************************/
#include <stdio.h>
#define NMAX 100

void leia_vetor(int v[], int n);
void imprima_vetor(int v[], int n);
int eh_alternante(int v[], int n);
void min_max(int v[], int n, int *pmin, int *pmax);

/************************************************************/
int main()
{
    int a[NMAX];
    int n, menor, maior;

    scanf("%d", &n);
    leia_vetor(a, n);  imprima_vetor(a, n);
    if (eh_alternante(a, n))
        printf("A sequencia dada e' alternante.\n");
    else
        printf("A sequencia dada nao e' alternante.\n");
    min_max(a, n, &menor, &maior);
    printf("Menor = %d   Maior = %d\n", menor, maior);
    return 0;
}

/************************************************************/
void leia_vetor(int v[], int n)
{
    int i;
    for (i = 0; i < n; i++)
        scanf("%d", &v[i]);
}

void imprima_vetor(int v[], int n)
{
    int i;
    printf("Os elementos da sequencia sao: ");
    for (i = 0; i < n; i++)
        printf("%d  ", v[i]);
    printf("\n");
}

int eh_alternante(int v[], int n)
{
    int i, p;
    for (i = 0; i < n - 1; i++) {
        p = v[i] + v[i + 1];
        printf("%d: %d\n", i, p);
        if (p % 2 == 0)
            return 0;
    }
    return 1;
}
```

```
void min_max(int v[], int n, int *pmin, int *pmax)
{
    int i;
    *pmin = v[0]; *pmax = v[0];
    for (i = 1; i < n; i++) {
        if (v[i] < *pmin) *pmin = v[i];
        if (v[i] > *pmax) *pmax = v[i];
        printf("[%d: %d %d]\n", i, *pmin, *pmax);
    }
}
/************************************************************/
```

**Questão 2** (valor: 3.0)

Dizemos que uma matriz inteira é *$K$-crescente* se essa matriz tem a seguinte propriedade: seus elementos são números inteiros positivos menores ou iguais a $K$, e estes crescem estritamente ao longo de cada *linha, da esquerda para a direita*. No exemplo abaixo temos uma matriz 50-crescente:

```
 4   7   8  10  17  28
 2   3   5   8  15  25
 8  10  14  16  20  48
 7   8   9  12  18  45
```

Note que cada linha está ordenada crescentemente *de maneira estrita*.

($a$) Escreva uma função em `C` de protótipo

```
int eh_crescente(int v[], int n);
```

que verifica se os `n` elementos do vetor fornecido `v[]` estão em ordem estritamente crescente, isto é, se

```
v[0] < v[1] < ... < v[n-1]
```

A sua função deve devolver `1` se `v[]` estiver em ordem estritamente crescente, e deve devolver `0` caso contrário.

(*b*) Escreva uma função em `C` de protótipo

```
int eh_K_crescente(int a[][NMAX], int m, int n, int K);
```

que verifica se a matriz `m x n` fornecida `a` é `K`-crescente. Neste item, *você deve usar a função* `eh_crescente()` *acima, mesmo que você não a tenha escrito.*

(*c*) Escreva um programa em `C` que recebe como entrada uma matriz $m \times n$ de inteiros $M$ e um inteiro $K$, e que determina se esta matriz é $K$-crescente. No caso de ela ser $K$-crescente, seu programa também deve determinar o menor $P$ para o qual esta matriz é $P$-crescente.

*Exemplo.* Suponha que o usuário forneça a entrada

```
4   6
2   3   5   8  15  25
4   7   8  10  17  28
8  10  14  16  20  48
7   8   9  12  18  45
50
```

(Note que $m$ e $n$ são fornecidos pelo usuário.) Então, o seu programa deve determinar que de fato a matriz dada é 50-crescente e deve também determinar que ela é também 48-crescente, mas não é 47-crescente. A saída deve ser algo como

```
Matriz eh 50-crescente.
Matriz eh tambem 48-crescente, mas nao eh 47-crescente.
```

Neste item, *você deve usar a função* `eh_K_crescente()` *acima, mesmo que você não a tenha escrito.*

**Questão 3** (valor: 4.0)

Um polinômio real $p(x)$ de grau no máximo $n$ é dado por

$$p(X) = p_0 + p_1 X + p_2 X^2 + \ldots + p_n X^n,$$

onde os coeficientes $p_0, p_1, \ldots, p_n$ são números reais. Se $p_n \neq 0$ então o grau de $p(X)$ é $n$.

Observe que um tal polinômio $p(X)$ pode ser representado por um vetor, digamos `p[0..n]` (naturalmente, o elemento `p[0]` do vetor contém o valor do coeficiente $p_0$, o elemento `p[1]` do vetor contém o valor do coeficiente $p_1$, e assim por diante).

($a$) Escreva uma função em `C` de protótipo

```
void leia_pol(double p[], int n);
```

que lê os coeficientes de um polinômio $p(X)$. Se o usuário quiser fornecer o polinômio $p(X) = 1 + 2X - X^3$, a entrada que ele fornecerá será

```
3
1 2 0 -1
```

Note que estamos supondo que o usuário fornecerá o grau do polinômio explicitamente (o inteiro `3` acima).

($b$) Escreva uma função em `C` de protótipo

```
void imprima_pol(double p[], int n);
```

que imprime os coeficientes do polinômio de grau `n` representado no vetor `p[]`. Se $p(X)$ é como no exemplo acima, a sua função deve imprimir algo como

```
grau: 3
coeficientes: 1 2 0 -1
```

(*c*) Escreva uma função em `C` de protótipo

```
double valor(double p[], int n, double x);
```

que recebe em `p[]` (os coeficientes) de um polinômio $p(X)$, em `n` o grau deste polinômio, e em `x` um número real, e que devolve o valor de $p(X)$ calculado neste número real.

> *Exemplo.* Suponha que $p(X) = 1 + 2X - X^3$ e que `x` $= 0.5$. Então a chamada da função `valor(p, n, x)`, onde `p[]` tem elementos `1`, `2`, `0`, e `-1`, `n = 3`, e `x = 0.5`, deve devolver o valor $p(0.5) = 1.875$.

*Observação.* Você *não pode usar* a função `pow()` nesta questão. Faça como temos feito em sala: para calcular a $i$-ésima potência de `x`, use o valor já determinado da $(i-1)$-ésima potência de `x`.

(*d*) Escreva um programa em `C` que recebe como entrada um polinômio $p(X)$, um inteiro $m$, e reais $x_1, \ldots, x_m$, e que imprime os valores $p(x_1), \ldots, p(x_m)$. Neste item, *você deve usar as funções dos itens anteriores, mesmo que você não as tenha feito.*

*Exemplo.* Suponha que $p(X) = 1 + 2X - X^3$, $m = 4$, e $x_1 = 0$, $x_2 = 0.5$, $x_3 = 1$, e $x_4 = 2$. A entrada para o seu programa deverá então ser

```
3
1 2 0 -1
4
0 0.5 1 2
```

e a saída deverá ser

```
grau: 3
coeficientes: 1 2 0 -1
p(0) = 1
p(0.5) = 1.875
p(1) = 2
p(2) = -3
```

*Observação.* Nesta questão, você deve supor que todos polinômios têm grau no máximo `999`.