Universidade Federal do ABC
BCM0505–15 — Processamento da Informação — Prática
Prova 2
Primeiro Quadrimestre de 2018

Nome:

RA:

Questão	Valor	Nota
1	2,0	
2	2,0	
3	3,0	
4	3,0	
Total	10,0	

Instruções:

- Em caso de fraude, **todos** os envolvidos receberão nota **zero**.
- Respostas às questões com erros de compilação receberão nota **zero**.

Boa prova!

1. Neste problema você deve calcular a soma dos elementos que estão na borda de uma matriz de N linhas e M colunas.

   A imagem abaixo ilustra os elementos que deverão ser considerados na operação em uma matriz de 6 linhas e 13 colunas (marcados com "X"):

X	X	X	X	X	X	X	X	X	X	X	X	X
X												X
X												X
X												X
X												X
X	X	X	X	X	X	X	X	X	X	X	X	X

   Entrada: O programa deve receber inicialmente dois inteiros N e M que indicam, nessa ordem, o número de linhas e colunas da matriz, onde $N \geq 2$ e $M \geq 2$. Em seguida deve receber $N \times M$ números reais que compõem a matriz, sendo que a mesma é preenchida linha por linha, da linha 0 até a linha $N - 1$, sempre da esquerda para a direita.

   Saída: A resposta consiste de uma única linha, contendo a soma dos elementos da borda da matriz, com 1 casa decimal.

ENTRADA	SAÍDA
4 3 3.2 5.0 8.1 0.2 -3.5 9.1 90.2 2.0 -13.0 0.0 3.5 10.0	116.3
2 5 3 7 5 3 9 9 0 7 1 5	49.0
2 2 1 0 0 -1	0.0

```java
import java.util.Scanner;
public class P2Q1{
    public static void main(String []args){
        Scanner tcl = new Scanner(System.in);
        int N = tcl.nextInt();
        int M = tcl.nextInt();
        double soma_borda = 0;
        for(int i=0;i<N;i++){
            for(int j=0;j<M;j++){
                /* Não é necessário armazenar a matriz, já que sabemos quais
                   valores queremos somar, val é equivalente a matriz[i][j] */
                double val = tcl.nextDouble();
                /*
                   Um elemento da borda tem, pelo menos, uma das seguintes
                   características:
                  - está na primeira linha (i == 0)
                  - está na primeira coluna (j == 0)
                  - está na última linha (i == N-1)
                  - está na última coluna (j == M-1) */
                if(i == 0 || j == 0 || i == (N-1) || j == (M-1)){
                    soma_borda += val;
                }
            }
        }
        System.out.printf("%.1f\n", soma_borda);
    }
}
```

2. Uma matriz quadrada de inteiros é um quadrado mágico se a soma dos elementos de cada linha, a soma dos elementos de cada coluna, a soma dos

elementos da diagonal principal e da diagonal secundária são todos iguais. A matriz abaixo é um exemplo de quadrado mágico:

$$\begin{bmatrix} 3 & 4 & 8 \\ 10 & 5 & 0 \\ 2 & 6 & 7 \end{bmatrix} \quad \Rightarrow \quad \begin{array}{cccccc} & & & & \nearrow & 15 \\ 3 & 4 & 8 & \rightarrow & & 15 \\ 10 & 5 & 0 & \rightarrow & & 15 \\ 2 & 6 & 7 & \rightarrow & & 15 \\ \downarrow & \downarrow & \downarrow & \searrow & & \\ 15 & 15 & 15 & & & 15 \end{array}$$

Faça um programa que lê uma matriz quadrada e determina se ela é um quadrado mágico.

Entrada: O programa deve receber inicialmente um inteiro N, com $N \geq 1$. Em seguida, deve receber $N \times N$ inteiros quaisquer que compõem a matriz, sendo que a mesma é preenchida linha por linha, da linha 0 até a linha N, sempre da esquerda para a direita.

Saída: A resposta consistirá de uma única linha, que deve conter a palavra `SIM` caso a matriz dada seja um quadrado mágico e `NAO` caso contrário.

ENTRADA	SAÍDA
3 3 4 8 10 5 0 2 6 7	SIM
4 4 14 15 1 9 7 6 12 5 11 10 8 16 2 3 13	SIM
3 3 4 8 10 5 1 2 6 7	NAO

```java
import java.util.Scanner;
public class P2Q2{
    public static void main(String []args){
        Scanner tcl = new Scanner(System.in);
        int N = tcl.nextInt();
        int []soma_linhas = new int[N];
        int []soma_colunas = new int[N];
        int soma_diag_princ = 0, soma_diag_sec = 0;
        boolean eh_quad_magico;
        for(int i=0;i<N;i++){
            for(int j=0;j<N;j++){
```

```
12                  /* Como já sabemos onde cada célula deve ser considerada não
13                     precisamos guardar a matriz, val é equivalente a matriz[i][j] */
14                  int val = tcl.nextInt();
15                  soma_linhas[i] += val;
16                  soma_colunas[j] += val;
17                  if(i == j) {
18                      soma_diag_princ += val;
19                  }
20                  if(j == N-i-1){
21                      soma_diag_sec += val;
22                  }
23              }
24          }
25          /* Basta que uma das condições não seja satisfeita para que não
26             seja um quadrado mágico */
27          eh_quad_magico = (soma_diag_princ == soma_diag_sec) &&
28                           (soma_linhas[0] == soma_diag_princ) &&
29                           (soma_colunas[0] == soma_diag_princ);
30          for(int i=1;i<N && eh_quad_magico;i++){
31              eh_quad_magico = (soma_linhas[i] == soma_colunas[i]) &&
32                               (soma_linhas[i-1] == soma_linhas[i]);
33          }
34          System.out.println(eh_quad_magico ? "SIM": "NAO");
35      }
36  }
```

---

3. O problema da ordenação é um dos mais básicos em computação, talvez sendo o que tenha o maior número de aplicações. Por exemplo, podemos ordenar uma fila de pessoas por ordem de idade para priorizar atendimento, ordenar os links de um resultado de busca por ordem de popularidade, ordenar uma lista de produtos por preço, ordenar uma lista de nomes por ordem alfabética, entre várias outras coisas.

   Considere que temos um vetor no qual estão armazenados n números inteiros, que desejamos, por requerimento da aplicação, ordenar de forma crescente. Uma ideia para fazer isso é a seguinte: encontre o menor elemento que esteja armazenado entre as posições 0 e $n-1$ (isto é, a partir da posição 0) e troque este elemento com o elemento que está na posição 0; encontre o menor elemento que está armazenado a partir da posição 1 (entre as posições 1 e $n-1$) e troque este elemento com o elemento que está na posição 1; encontre o menor elemento que está armazenado a partir da posição 2 e troque este elemento com o que está na posição 2; e assim sucessivamente...

   No exemplo abaixo, os elementos sublinhados representam os elementos

que serão trocados na i-ésima iteração do algoritmo:

// Elemento 11 é o menor entre as posições 0 e 5.
// Troque com o elemento 57 (que está na posição 0):
iteração 0 = (57, 32, 25, 11, 90, 63)
// Elemento 25 é o menor entre as posições 1 e 5.
// Troque com o elemento 32 (que está na posição 1):
iteração 1 = (11, 32, 25, 57, 90, 63)
// Elemento 32 é o menor entre as posições 2 e 5.
// Troque com o elemento 32 (que está na posição 2):
iteração 2 = (11, 25, 32, 57, 90, 63)
// Elemento 57 é o menor entre as posições 3 e 5.
// Troque com o elemento 57 (que está na posição 3):
iteração 3 = (11, 25, 32, 57, 90, 63)
// Elemento 63 é o menor entre as posições 4 e 5.
// Troque com o elemento 90 (que está na posição 4):
iteração 4 = (11, 25, 32, 57, 90, 63)
// Elemento 90 é o menor entre as posições 5 e 5.
// Troque com o elemento 90 (que está na posição 5):
iteração 5 = (11, 25, 32, 57, 63, 90)

Faça um programa que implemente o algoritmo explicado acima.

Entrada: O programa deverá receber inicialmente um valor inteiro n, com $n > 1$. Em seguida, deverá receber n inteiros **quaisquer**.

Saída: A saída consistirá de uma única linha, contendo os n inteiros que foram recebidos na entrada em ordem crescente e separados por um espaço. Veja os exemplos.

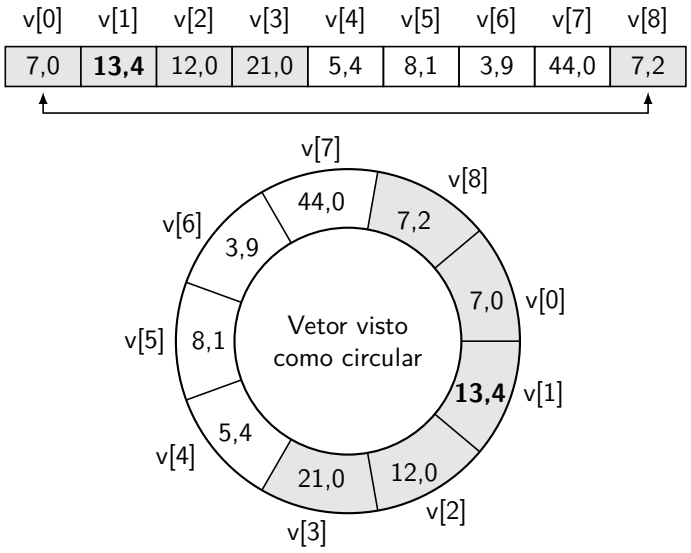
ENTRADA	SAÍDA
6 57 32 25 11 90 63	11 25 32 57 63 90
7 -8 4 2 9 1 7 90	-8 1 2 4 7 9 90

```java
import java.util.Scanner;
public class P2Q3{
    public static void main(String []args){
        Scanner tcl = new Scanner(System.in);
        int N = tcl.nextInt();
        int []nums = new int[N];
        for(int i=0;i<N;i++){
            nums[i] = tcl.nextInt();
        }
        for(int i=0;i<N-1;i++){
            /* Encontrar o indíce do menor elemento entre as
                posições i e N-1  */
```

```
13              int ind_menor = i;
14              for(int j=i+1;j<N;j++){
15                  if(nums[j] < nums[ind_menor]){
16                      ind_menor = j;
17                  }
18              }
19              int tmp = nums[i];
20              nums[i] = nums[ind_menor];
21              nums[ind_menor] = tmp;
22          }
23
24          for(int i=0;i<N;i++){
25              System.out.printf("%d ",nums[i]);
26          }
27          System.out.println();
28      }
29  }
```

4. Escreva um programa que recebe pela entrada padrão N, o número de amostras, seguido dos N valores (`float`) das amostras. Então o seu programa deverá ler A inteiro, $0 \leq A < N$ que indica a posição sobre a qual desejamos calcular a média e J, $1 \leq J < \frac{N}{2}$ que indica a janela sobre a qual a média deverá ser calculada. Caso a janela seja maior (tanto do lado esquerdo quanto do lado direito) do que os limites do vetor, o seu programa deverá tratar o vetor como sendo circular, ou seja, como se o início do vetor estivesse conectado ao seu fim.

   Exemplo: $N = 9$, $A = 1$, $J = 2$



.

ENTRADA	SAÍDA
9 7.0 13.4 12.0 21.0 5.4 8.1 3.9 44.0 7.2 1 2	12.2

```java
import java.util.Scanner;
public class P2Q4{
    public static void main(String []args){
        Scanner tcl = new Scanner(System.in);
        float media = 0;
        int N = tcl.nextInt();
        float[] nums = new float[N];
        for(int i=0;i<N;i++){
            nums[i] = tcl.nextFloat();
        }
        int A = tcl.nextInt();
        int J = tcl.nextInt();

        // Encontrando a posição do primeiro elemento a ser somado
        int pos = A - J;
        if(pos < 0){
            pos = N + pos;
        }
        //J*2+1 é o número de elementos que devem ser somados
        for(int i=0; i < J*2+1; i++){
            media += nums[pos];
            /* Ao percorrermos o vetor devemos considerar a circularidade
               por isso o módulo */
            pos = (pos + 1) % N;
        }
        media /= J*2+1;
        System.out.printf("%.1f\n", media);
    }
}
```