AG22S - Algoritmos 2

Matrizes

4^a Prática

Nome: _____ Turma: _____ Código: _____ Nº: ____

Número de Problemas Corretos	Nota
0	0.0
1	3.0
2	6.0
3+	10.0

1. Faça um programa que lê da entrada padrão um inteiro N que corresponde ao tamanho da matriz. Em seguida, o programa deverá ler uma matriz quadrada, M, de tamanho N x N. O programa deve imprimir a adição de matrizes M + M, como mostrado na figura abaixo.

$$\begin{bmatrix} a_1 & b_1 \\ c_1 & d_1 \end{bmatrix} + \begin{bmatrix} a_2 & b_2 \\ c_2 & d_2 \end{bmatrix} = \begin{bmatrix} a_1 + a_2 & b_1 + b_2 \\ c_1 + c_2 & d_1 + d_2 \end{bmatrix}$$

Entrada

A entrada inicia com uma linha contendo o número inteiro N ($0 < N \leq 10$) seguido de N linhas. Cada uma das linhas descreve uma linha da matriz e contendo N números reais (double).

Saída

Imprima a matriz de resposta em N linhas, cada linha deve imprimir cada elemento (número) separado por um espaço em branco. Use uma casa decimal.

Exemplo de entrada	**Exemplo de saída**

`3`
`1.0 2.0 3.0`
`4.0 5.0 6.0`
`7.0 8.0 9.0`
	`2.0 4.0 6.0`
	`8.0 10.0 12.0`
	`14.0 16.0 18.0`

Dica

Para que a resposta esteja correta no programa de correção automática, não pode haver espaços em branco no final das linhas.

2. Faça um programa que lê da entrada padrão um inteiro N que corresponde ao tamanho da matriz. Em seguida, o programa deverá ler uma matriz quadrada, M, de tamanho N x N. O programa deve imprimir a multiplicação de matrizes M * M, como mostrado na figura abaixo.

$$\begin{bmatrix} a_1 & b_1 \\ c_1 & d_1 \end{bmatrix} \times \begin{bmatrix} a_2 & b_2 \\ c_2 & d_2 \end{bmatrix} = \begin{bmatrix} a_1a_2 + b_1c_2 & a_1b_2 + b_1d_2 \\ c_1a_2 + d_1c_2 & c_1b_2 + d_1d_2 \end{bmatrix}$$

Entrada

A entrada inicia com uma linha contendo o número inteiro N ($0 < N \leq 10$) seguido de N linhas. Cada uma das linhas descreve uma linha da matriz e contendo N números reais (double).

Saída

Imprima a matriz de resposta em N linhas, cada linha deve imprimir cada elemento (número) separado por um espaço em branco. **Use uma casa decimal**.

Note que no primeiro exemplo temos:

$1 * 1 + 2 * 4 = 9.0$

$1 * 2 + 2 * 5 = 12.0$

$4 * 1 + 5 * 4 = 24.0$

$4 * 2 + 5 * 5 = 33.0$

Exemplo de entrada	**Exemplo de saída**
2 1 2 4 5	9.0 12.0 24.0 33.0
Exemplo de entrada	**Exemplo de saída**
3 1 2 3 4 5 6 7 8 9	30.0 36.0 42.0 66.0 81.0 96.0 102.0 126.0 150.0

3. Faça um código que leia uma matriz 3 x 3 e imprima a matriz transposta. Para imprimir a matriz transposta, troque as linhas pelas colunas conforme exemplo de entrada e saída.

   Entrada

   O caso de teste é composto por três linhas. Cada linha possui três inteiros descrevendo uma linha da matriz.

   Saída

   Imprima três linhas, cada uma contendo três inteiros (`int`) separados por um espaço em branco, cada linha é a linha da matriz após a troca.

Exemplo de entrada	**Exemplo de saída**
1 2 3 4 5 6 7 8 9	1 4 7 2 5 8 3 6 9

4. Faça um código que receba um número entre 1 e 12 e escreva o mês correspondente. O seu código deve declarar e utilizar a matriz abaixo:

```
char mes[12][4] = {"jan", "fev", "mar", "abr",
    "mai", "jun", "jul", "ago", "set", "out",
    "nov", "dez"};
```

   Entrada

   O caso de teste é composto por uma linha contendo um número $0 < N \leq 12$.

   Saída

   Imprima uma linha com o nome do mês conforme definido na matriz `mes`

Exemplo de entrada	**Exemplo de saída**
1	jan
Exemplo de entrada	**Exemplo de saída**
7	jul

5. Faça um código que receba N nomes e os armazene em uma matriz de caracteres, onde cada linha irá armazenar um nome. Seu programa então receberá um índice x, mostre uma linha com o nome guardado naquele índice da matriz.

   Entrada

   A entrada inicia com uma linha contendo o número, $0 < N \leq 10$, de nomes. Cada uma das próximas N linhas contém uma string de até 32 caracteres contendo um nome. Finalmente, seu programa recebe um índice $0 < x < N$.

   Saída

   Imprima uma linha com o nome guardado no indice indicado.

Exemplo de entrada	**Exemplo de saída**

`3`
`Fulano de tal`
`Beltrano`
`Sicrano`
`1`
	`Beltrano`

Desafios

Os problemas desta parte são extras, avançados e não são contabilizados na nota. O professor não dará suporte durante a aula na resolução destes problemas.

1. **VARETAS - Jogo de Varetas**

   Há muitos jogos divertidos que usam pequenas varetas coloridas. A variante usada neste pro- blema envolve a construção de retângulos. O jogo consiste em, dado um conjunto de varetas de comprimentos variados, desenhar retângulos no chão, utilizando as varetas como lados dos retângulos, sendo que cada vareta pode ser utilizada em apenas um retângulo, e cada lado de um retângulo é formado por uma única vareta. Nesse jogo, duas crianças recebem dois conjuntos iguais de varetas. Ganha o jogo a criança que desenhar o maior número de retângulos com o conjunto de varetas.

   Dado um conjunto de varetas de comprimentos inteiros, você deve escrever um programa para determinar o maior número de retângulos que é possível desenhar.

   Entrada

   A entrada contém vários casos de teste. A primeira linha de um caso de teste contém um inteiro N que indica o número de diferentes comprimentos de varetas ($1 \leq N \leq 1.000$) no conjunto. Cada uma das N linhas seguintes contém dois números inteiros C_i e V_i, representando respectivamente um comprimento ($1 \leq C_i \leq 10.000$) e o número de varetas com esse comprimento ($1 \leq V_i \leq 1.000$). Cada comprimento de vareta aparece no máximo uma vez em um conjunto de teste (ou seja, os valores C_i são distintos). O final da entrada é indicado por $N = 0$.

   A entrada deve ser lida da entrada padrão.

   Saída

   Para cada caso de teste da entrada seu programa deve produzir uma única linha na saída, contendo um número inteiro, indicando o número máximo de retângulos que podem ser formados com o conjunto de varetas dado.

   A saída deve ser escrita na saída padrão.

Exemplo de entrada	**Exemplo de saída**
1 10 7 4 50 2 40 2 30 4 60 4 5 15 3 6 3 12 3 70 5 71 1 0	1 3 2

2. **HIST - Histórico de Comandos**

Uma interface por linha de comando (ILC) é um dos tipos de interface humano-computador mais antigos que existem. Uma ILC permite a interação com o software através de um interpretador de comandos, sendo normalmente acessível em um terminal (ou janela) de texto. A vantagem de um interpretador de comandos é que ele permite que o usuário opere o sistema usando apenas o teclado. Ainda hoje em dia, em que estamos acostumados com interfaces gráficas sofisticadas, muitos aplicativos e sistemas operacionais incluem algum tipo de interface por linha de comando, e muitos usuários ainda preferem usá-la para grande parte das tarefas.

Um dos recursos mais úteis de um interpretador de comandos é o histórico de comandos. Quando um comando é digitado e executado, ele é colocado no histórico de comandos do terminal. O comando pode ser exibido novamente no terminal apertando a tecla '↑'; a tecla Enter executa o comando novamente quando o comando está sendo exibido no terminal. Todos os comandos executados são guardados no histórico: pressionar a tecla '↑' duas vezes exibe o penúltimo comando executado, pressioná-la três vezes exibe o antepenúltimo comando, e assim sucessivamente.

Por exemplo, se o histórico inicial é (A, B, C, D), para repetir o comando C basta pressionar duas vezes a tecla '↑'. O histórico será então atualizado para (A, B, C, D, C). Nesse ponto, para repetir o comando A será necessário pressionar cinco vezes a tecla '↑'; o histórico será atualizado para (A, B, C, D, C, A). Nesse ponto, para repetir mais uma vez o comando A basta pressionar uma vez a tecla '↑'; o histórico será atualizado para (A, B, C, D, C, A, A).

Leandro é administrador de sistemas e usa frequentemente o interpretador de comandos para gerenciar remotamente os servidores que administra. Em geral, ele precisa apenas repetir comandos que já havia digitado antes. Enquanto estava trabalhando em um servidor, ele teve uma curiosidade: quantas vezes ele precisa pressionar a tecla '↑' para executar uma determinada sequencia de comandos? Ele sabe quais são as posições no histórico dos comandos que ele necessita executar, mas não sabe resolver esse problema. Por isso, pediu que você fizesse um programa que respondesse à pergunta dele.

Entrada

A entrada é composta de vários casos de teste. A primeira linha de cada caso de teste contém um número inteiro N, indicando o número de comandos que Leandro deseja executar ($1 \leq N \leq 1.000$). A segunda linha de um caso de teste contém N inteiros $P_1, P_2, \ldots, P_N$, que indicam as posições dos comandos no histórico ($1 \leq Pi \leq 1.000.000$) no momento inicial, na ordem em que os comandos devem ser executados. Ou seja, o primeiro comando que deve ser executado está inicialmente na posição P_1 do histórico; depois deve ser executado o comando que está inicialmente na posição P_2 no histórico, e assim por diante, até P_N, que é a posição inicial do último comando que deve ser executado. Note que pode haver $P_i = P_j$

As posições são dadas em função do número de vezes que a tecla '↑' deve ser pressionada: um comando na posição 5 necessita que a tecla '↑' seja pressionada cinco vezes antes de aparecer no terminal (note que à medida que comandos vão sendo executados, a posição de um dado comando no histórico pode mudar).

O final da entrada é indicado por N = 0.

A entrada deve ser lida da entrada padrão.

Saída

Para cada caso de teste, seu programa deve imprimir apenas uma linha, contendo o número de vezes que Leandro precisa pressionar a tecla '↑' para executar todos os comandos.

A saída deve ser escrita na saída padrão.

Exemplo de entrada	Exemplo de saída
3 2 5 3 4 2 1 4 3 5 1 2 3 4 5 4 1 3 1 3 0	13 16 25 9