CENTRO PAULA SOUZA
COMPETÊNCIA EM EDUCAÇÃO PÚBLICA PROFISSIONAL
Centro Estadual de Educação Tecnológica Paula Souza
GOVERNO DO ESTADO DE SÃO PAULO

VII Maratona de Programação da FATEC-Rubens Lara

19 de novembro de 2016

Caderno de Problemas

Este caderno contém 6 problemas, as páginas estão numeradas de 1 a 8, não contando esta página de rosto.

Informações Gerais:

A) Sobre a entrada

1) A entrada de seu programa deve ser lida da entrada padrão.

2) A entrada é composta por um único caso de teste, descrito em um número de linhas que depende do problema.

3) Quando uma linha da entrada contém vários valores, estes são separados por um único espaço em branco; a entrada não contém nenhum outro espaço em branco.

4) Cada linha, incluindo a última, contém o caractere final de linha.

B) Sobre a saída

1) A saída de seu programa deve ser escrita na saída padrão.

2) Quando uma linha da saída contém vários valores, estes devem ser separados por um único espaço em branco; a saída não deve conter nenhum outro espaço em branco.

3) Cada linha, incluindo a última, deve conter o caractere final de linha.

Realização:

FATEC BS

Problema A
Zipper

Nome do arquivo fonte: zipper. [c | cpp | java | hs]

Em programação funcional pura, um Zipper é uma técnica que modifica uma estrutura de dados recursiva tais como listas e árvores, no intuito de ajudar no percurso de tais estruturas. Uma das formas de um Zipper para listas, funciona da seguinte maneira: pegue uma lista qualquer, por exemplo, [1,2,3,4,5]. Se quisermos descrever a localização do 2 usando um Zipper obteremos, [1,2] 3 [4,5]. No exemplo, 2 é o foco do Zipper e as duas listas representam os caminhos pela esquerda e pela direita possíveis caso queiramos percorrer a lista. Caso quisermos ir para a esquerda de 2 teremos, [1] 2 [3,4,5] e caso quisermos ir para a esquerda novamente teremos [] 1 [2,3,4,5] e neste caso não há mais possibilidade de ida nesta mesma direção.

Sua tarefa é implementar um programa que simule um caminho em uma lista com um Zipper.

Entrada

A primeira linha da entrada é composta por um inteiro N ($2 < N < 100$), representando o número de elementos da lista. A segunda linha é composta por N inteiros X_i separados por um espaço em branco ($-1000 \leq X_i \leq 1000$, com $0 \leq i \leq N-1$). A terceira contém dois inteiros I ($0 \leq I \leq N-1$) e C ($1 \leq C \leq 100$) separados por um espaço, representando o índice do primeiro foco e o comprimento do percurso do Zipper, respectivamente. As próximas C linhas conterão as palavras `esq` ou `dir` indicando o percurso do Zipper.

Saída

Seu programa deve produzir uma linha contendo a configuração da lista com o Zipper resultante no seguinte formato: os elementos a esquerda do foco entre colchetes e separados por vírgulas, seguida do elemento do foco, seguida dos elementos a direita do foco entre colchetes separados por vírgulas.

Exemplos

Entrada	**Saída**
10 1 9 8 3 4 6 10 -1 9 10 7 9 esq dir dir dir esq dir dir dir	[1,9,8,3,4,6,10,-1,9]10[]

Entrada	**Saída**
5 1 2 3 4 5 1 3 esq esq dir	[1]2[3,4,5]

Problema B

Eleição

Nome do arquivo fonte: eleicao. [c | cpp | java | hs]

Na eleição Americana, sabe-se que o voto popular não elege seu presidente e sim o colégio eleitoral. A eleição deve ser ganha por estados e cada estado possui um peso chamado *electoral vote* calculado com base em sua população. Com exceção dos estados do Maine e Nebraska, a eleição procede da seguinte forma: se um candidato ganha em um estado, esse possui todos os votos do colégio (*electoral votes*).

Deixemos de lado, aqui, as exceções. Em uma Universidade, o mesmo sistema será adotado para a eleição de seu Reitor com os estados sendo representados pelos alunos/professores matriculados por instituto. Sua tarefa é implementar um programa que apresente o resultado final das eleições.

Entrada

A primeira linha da entrada é composta por dois inteiros, I ($1 < I < 100$) e C ($2 \leq C \leq 5$), indicando o número de institutos (estados) e o número de candidatos, respectivamente. A próxima linha é composta por I inteiros E_k ($1 \leq E_k \leq 10^4$ para $1 \leq k \leq I$), indicando a quantidade de eleitores de cada estado. A terceira linha contém I inteiros P_k ($3 \leq P_k \leq 60$ para $1 \leq k \leq I$), representando a quantidade de votos (*Electoral Votes*) adquiridos pelo vencedor em cada instituto. A quarta linha contém uma string representando a vontade da universidade, sendo que o caractere 0 indica nulo, '!' representa o candidato A, '@' o B, '#' o C, '$' o D e '%' o E que foram lidos das urnas.

Saída

Seu programa deve produzir C linhas, representando o resultado final indicando o número de votos (*Electoral Votes*) que cada candidato recebeu. Cada linha contém um caractere que representa o candidato ('A', 'B', 'C', 'D' ou 'E'), seguido por dois pontos (':'), seguido por um espaço em branco, seguido pelo número de votos do candidato.

Exemplo

Entrada	**Saída**
3 3 10 15 35 5 12 16 !!!00@@0!!!!!@!@@@@@!!@@00000!!!!!@!@@@@!!!@@!@!@!@###@@###@	A: 5 B: 28 C: 0

Problema C
Sudoku

Nome do arquivo fonte: sudoku. [c | cpp | java | hs]

O jogo de Sudoku espalhou-se rapidamente por todo o mundo, tornando-se hoje o passatempo mais popular em todo o planeta. Muitas pessoas, entretanto, preenchem a matriz de forma incorreta, desrespeitando as restrições do jogo. Sua tarefa neste problema é escrever um programa que verifica se uma matriz preenchida é ou não uma solução para o problema.

A matriz do jogo é uma matriz de inteiros 9 x 9 . Para ser uma solução do problema, cada linha e coluna deve conter todos os números de 1 a 9. Além disso, se dividirmos a matriz em 9 regiões 3 x 3, cada uma destas regiões também deve conter os números de 1 a 9. O exemplo abaixo mostra uma matriz que é uma solução do problema.

$$\left(\begin{array}{ccc|ccc|ccc} 1 & 3 & 2 & 5 & 7 & 9 & 4 & 6 & 8 \\ 4 & 9 & 8 & 2 & 6 & 1 & 3 & 7 & 5 \\ 7 & 5 & 6 & 3 & 8 & 4 & 2 & 1 & 9 \\ \hline 6 & 4 & 3 & 1 & 5 & 8 & 7 & 9 & 2 \\ 5 & 2 & 1 & 7 & 9 & 3 & 8 & 4 & 6 \\ 9 & 8 & 7 & 4 & 2 & 6 & 5 & 3 & 1 \\ \hline 2 & 1 & 4 & 9 & 3 & 5 & 6 & 8 & 7 \\ 3 & 6 & 5 & 8 & 1 & 7 & 9 & 2 & 4 \\ 8 & 7 & 9 & 6 & 4 & 2 & 1 & 5 & 3 \end{array}\right)$$

Entrada

A entrada contém 9 linhas, em que cada linha contém 9 números inteiros de 1 a 9 representando uma linha da matriz do Sudoku.

Saída

Seu programa deverá imprimir "*SIM*" se a matriz for a solução de um problema de Sudoku, e "*NAO*" caso contrário.

Exemplos

Entrada	**Saída**
1 3 2 5 7 9 4 6 8 4 9 8 2 6 1 3 7 5 7 5 6 3 8 4 2 1 9 6 4 3 1 5 8 7 9 2 5 2 1 7 9 3 8 4 6 9 8 7 4 2 6 5 3 1 2 1 4 9 3 5 6 8 7 3 6 5 8 1 7 9 2 4 8 7 9 6 4 2 1 5 3	SIM

Entrada	**Saída**
1 3 2 5 7 9 4 6 8 4 9 8 2 6 1 3 7 5 7 5 6 3 8 4 2 1 9 6 4 3 1 5 8 7 9 2 5 2 1 7 9 3 8 4 6 9 8 7 4 2 6 5 3 1 2 1 4 9 3 5 6 8 7 3 6 5 8 1 7 9 2 4 8 7 9 6 4 2 1 3 5	NAO

Problema D

Caça Palavras

Nome do arquivo fonte: palavras. [c | cpp | java | hs]

O pequeno Ivo está aprendendo suas primeiras palavras. Com a ajuda de seus pais e amigos, ele aprendeu a jogar caça palavras para praticar e adquirir novo vocabulário. Um jogo de caça palavras consiste em um arranjo retangular de letras minúsculas e uma lista de palavras (também em letras minúsculas), com o objetivo de encontrar o maior número possível de palavras da lista no arranjo de letras. As palavras podem aparecer na horizontal e na vertical e na ordem correta ou inversa.

Os pais e amigos de Ivo criaram alguns jogos de caça palavras para que ele possa se divertir. Eles estão pedindo ajuda para verificar se as respostas de Ivo estão corretas. Você pode ajudá-los?

Entrada

A primeira linha contém 3 inteiros L, C e P ($1 \leq L, C, P \leq 100$), representando respectivamente o número de linhas e colunas do retângulo com o arranjo de letras e número de palavras a procurar no jogo. Cada uma das L linhas seguintes consiste em exatamente C letras minúsculas. Depois, as próximas P linhas descrevem a lista de palavras para procurar no arranjo de letras. Cadas uma dessas linhas contém uma sequência não vazia de, no máximo, 100 letras minúsculas.

Saída

Seu programa deve produzir uma única linha contendo o número máximo de palavras da lista de P palavras que aparecem no arranjo retangular de L x C letras.

Exemplos

Entrada	**Saída**
8 11 3 adhjdnfdjfb nwjsafksoat ogfdgkfdftx tdgfdgldnec afgfgdsqwkc rfsantosoid afdgfdgfmao mfgfdgfdqaz maratona fatec santos	2

Entrada	**Saída**
2 2 6 cd ba ba dc cb ad da ac	5

Problema E

Driblando os Pedágios

Nome do arquivo fonte: pedagios. [c | cpp | java | hs]

Bino é um caminhoneiro experiente, ele e seu companheiro Pedro já transportaram muita carga pesada por esse Brasil afora. Mas atualmente Bino anda indignado com o preço dos pedágios nas estradas paulistas, pois em muitas situações o preço do frete não compensa o transporte da carga devido ao alto custo dos pedágios.

Bino pediu sua ajuda para escrever um programa que dado um mapa com a distância e o custo com pedágios de um conjunto de estradas que interligam as cidades do estado, determine dois tipos de caminhos entre duas cidades do mapa:

a) o que possui o menor custo com pedágio;

b) o que possui a menor distância.

Eventualmente estes dois caminhos podem ser os mesmos. As estradas tem sentido duplo, ou seja, se há uma estrada ligando as cidades A e B também há uma estrada ligando B a A com mesma distância e custo com pedágios.

Entrada

A primeira linha da entrada contém dois inteiros C ($2 \leq C \leq 100$) e E ($2 \leq E \leq C^2$), indicando o número de cidade e estradas do mapa, respectivamente. Cada uma das E linhas seguintes descreve uma estrada ligando duas cidades e contém quatro inteiros. A linha i, para $1 \leq i \leq E$, contém quatro inteiros X_i, Y_i, D_i e P_i, indicando que a cidade X_i está ligada a cidade Y_i por uma estrada com distância D_i e custo com pedágio P_i ($0 \leq X_i, Y_i \leq C$-1, $X_i \neq Y_i$, $1 \leq D_i \leq 10^3$ e $0 \leq P_i \leq 10^2$). A linha seguinte contém um inteiro N indicando o número de consultas que serão feitas ($1 \leq N < 10^3$). Cada uma das N linhas seguintes contém dois inteiros diferentes O e D ($0 \leq O, D \leq C$-1), representando as cidades de origem e destino.

Saída

Seu programa deve produzir 4 linhas para cada uma das C consultas da entrada. A primeira linha da saída referente a uma consulta contém dois inteiros D_p e P_p, representando a distância e o custo com pedágios, respectivamente, do caminho com o menor custo com pedágios. A segunda linha contém a lista das cidades separadas por espaço pertencentes ao caminho com o menor custo com pedágios. A terceira linha contém dois inteiros D_d e P_d, representando a distância e o custo com pedágios, respectivamente, do caminho com a menor distância. A quarta linha da saída de uma consulta contém a lista das cidades separadas por espaço pertencentes ao caminho com a menor distância.

Exemplos

Entrada	Saída
4 5 0 1 100 10 1 2 80 5 2 3 140 15 0 2 200 0 1 3 120 17 1 0 3	340 15 0 2 3 220 27 0 1 3

Entrada	**Saída**
6 8 0 1 60 15 0 3 220 5 1 2 70 5 1 3 90 12 2 3 50 4 2 5 150 10 3 4 80 20 4 5 75 0 2 0 4 1 5	495 19 0 3 2 5 4 230 47 0 1 3 4 220 15 1 2 5 220 15 1 2 5

Problema F

Cadernos

Nome do arquivo fonte: cadernos. [c | cpp | java | hs]

Joãozinho está comprando os cadernos para o seu próximo curso: Análise de Algoritmos. Joãozinho sabe exatamente quantas páginas são necessárias para cada aula durante o semestre. Tendo em vista a crise atual, Joãozinho só passa para um novo caderno quando todas as folhas do caderno atual tiverem sido preenchidas.

Joãozinho sempre compra cadernos com exatamente o mesmo número de páginas. Dadas quantas páginas são necessárias para cada uma das aulas do semestre, quantos cadernos Joãozinho precisa comprar para fazer as anotações de todas as aulas?

Entrada

A entrada contém duas linhas. A primeira linha da entrada contém dois inteiros N ($1 \leq N \leq 1000$), que é o número de aulas no semestre, e P ($1 \leq P \leq 1000$), que é o número de página de cada caderno. A segunda linha contém N inteiros A_1, A_2, …, A_N ($1 \leq A_i \leq P$, para $1 \leq i \leq N$), onde A_i é o número de páginas necessárias para a i-ésima aula.

Saída

Seu programa deve produzir uma linha contendo o número de cadernos que Joãozinho precisa para todo o semestre.

Exemplos

Entrada	**Saída**
`4 100` `60 30 20 60`	`2`

Entrada	**Saída**
`3 100` `70 100 35`	`3`