Universidade Federal do Espírito Santo
Departamento de Informática
Programa de Educação Tutorial – PET EngComp
E-mail: petengcomp@inf.ufes.br
Home-Page: www.inf.ufes.br/~pet
Tel. (27) 3335-2161

Topcom 5

Torneio de Programação de Computadores

PROVA

Realização: Apoio: Patrocínio:

PROBLEMA A
ACHANDO PALAVRAS

Este problema é baseado em http://acm.uva.es/p/v8/892.html e adaptado pelo Prof. Dr. João Paulo Almeida Andrade, do Departamento de Informática da UFES.

Um problema comum ao se processar texto é a detecção das palavras no texto, excluindo-se os elementos da pontuação: vírgulas, "aspas" e (parênteses) anexos às palavras ou hífens do meio das palavras (além de exclamações, dígitos, pontos, dois pontos, etc.).

Faça um programa que "limpe" o texto de elementos da pontuação, isto é, que gere um texto baseado na entrada sem caracteres que não sejam letras de "A" a "Z" ou "a" a "z" ou espaços em branco.

A ENTRADA

Cada caso de teste da entrada consiste de linhas de no máximo 60 caracteres por linha. Cada linha é terminada por um caracter de retorno de linha. Um teste é terminado por uma linha que consiste em um único caracter "#", seguido por um caracter de retorno de linha. (A entrada não terá caracteres com acentuação).

A entrada conterá um ou mais casos de testes. Uma linha em branco separa dois testes distintos. A entrada é terminada por um fim de arquivo.

A SAÍDA

A saída deve conter linhas do texto de entrada, com os caracteres que não sejam letras removidos (com a exceção de espaços em branco). Há uma regra especial quando um hífen é usado para último caracter em uma linha: a palavra quebrada pelo hífen como último caracter em uma linha deve aparecer em uma linha apenas para a palavra. (Veja no exemplo abaixo o tratamento para "hy-phens"). As saídas de dois testes são separadas por uma linha em branca.

EXEMPLO DE ENTRADA

```
A common problem when processing incoming text is to isolate
the words in the text.  This is made more difficult by the
punctuation; words have commas, "quote marks",
(even brackets)      next to them, or hy-
phens in the middle of the word.  This punctuation does not
count as letters when the words have to be looked up in a
# dictionary by the 12345 "**&! program.
#

The second test will be place here, after a blank line!
#
```

EXEMPLO DE SAÍDA

```
A common problem when processing incoming text is to isolate
```

```
the words in the text  This is made more difficult by the
punctuation words have commas quote marks
even brackets      next to them or
hyphens
 in the middle of the word  This punctuation does not
count as letters when the words have to be looked up in a
 dictionary by the   program

The second test will be place here after a blank line
```

PROBLEMA B
CHOCOLATE

A fábrica Fishburg produz chocolates na forma de polígonos convexos. Kiddy and Carlson compraram um e querem quebrá-lo em dois fragmentos. As áreas destes fragmentos devem ser iguais. Crie um código para encontrar o comprimento da menor linha de separação usando a forma do chocolate dada.

A ENTRADA

A primeira linha da entrada contém um inteiro N – o número de vértices do polígono ($3 < N \leq 50$). As outras N linhas contém as coordenadas dos vértices indo consecutivamente pelo contorno do polígono, girando no sentido horário. As coordenadas são números reais de -100 a 100.

A entrada conterá um ou mais casos de testes. A entrada é terminada por um fim de arquivo.

A SAÍDA

Para cada teste, a saída deve ser uma linha contendo o mínimo comprimento da linha de separação com precisão de 0.0001.

EXEMPLO DE ENTRADA

```
4
0  0
0  3
4  3
4  0
3
0  0
2  2
4  0
```

EXEMPLO DE SAÍDA

```
3
2
```

PROBLEMA C
INVERSÃO-K

Considere a permutação $a_1, a_2, \ldots, a_n$ (todos os a_i são diferentes inteiros que variam de 1 a n). Vamos chamar de inversão-k a seqüência de números de $i_1, i_2, ..., i_k$ tal que $1 \leq i_1 < i_2 < ... < i_k \leq n$ e $a_{i1} > a_{i2} > \ldots > a_{ik}$. Sua tarefa é calcular o número de inversões-k diferentes na permutação dada.

A ENTRADA

A primeira linha de cada teste contém dois inteiros n e k ($1 \leq n \leq 20000$, $2 \leq k \leq 10$). A segunda linha é preenchida com os n números a_i.

A entrada conterá um ou mais casos de testes. Uma linha em branco separa dois testes distintos. A entrada é terminada por um fim de arquivo.

A SAÍDA

Para cada teste imprima um número simples – o número de inversões-k numa dada permutação. Os resultados de dois casos de teste distintos são separados por uma linha em branco.

EXEMPLO DE ENTRADA

```
3 2
3 1 2

5 3
5 4 3 2 1
```

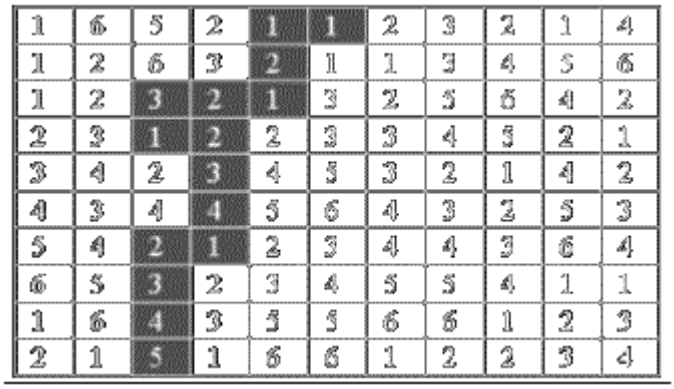
EXEMPLO DE SAÍDA

```
2

10
```

PROBLEMA D
LABIRINTO NUMÉRICO

Ninguém sabe quem é o criador do primeiro labirinto, mas todas as diversidades de labirintos podem ser encontradas em qualquer lugar e muitos destes foram construídos há muito tempo. O mais famoso labirinto é talvez um em Creta, desenhado e construído por Daedalus para o rei Minos, um lugar em que se acredita que o Minotauro (um monstro com a cabeça de touro e o corpo de homem) foi mantido, alimentado de carne humana, até ser destruído por Theseus. A seguinte figura ilustra um tipo de labirinto.



1	6	5	2	1	1	2	3	2	1	4
1	2	6	3	2	1	1	3	4	5	6
1	2	3	2	1	3	2	5	6	4	2
2	3	1	2	2	3	3	4	5	2	1
3	4	2	3	4	5	3	2	1	4	2
4	3	4	4	5	6	4	3	2	5	3
5	4	2	1	2	3	4	4	3	6	4
6	5	3	2	3	4	5	5	4	1	1
1	6	4	3	5	5	6	6	1	2	3
2	1	5	1	6	6	1	2	2	3	4

Você está preparado para encontrar um caminho que te leve da entrada à saída do labirinto? Você só é liberado para se mover horizontalmente ou verticalmente. Movimentos em diagonal não estão permitidos. Um caminho consiste de subseqüências obtidas da seguinte regra: 1; 1,2; 1,2,3; 1,2,3,4; e assim por diante. As subseqüências podem incluir mudanças na direção.

O problema que você tem que resolver é determinar uma entrada e um caminho que te leve a um ponto de saída, para um dado labirinto. O ponto de partida é sempre uma célula no topo do labirinto (com valor 1!) e o ponto de saída é uma célula na última linha do labirinto.

A ENTRADA

Uma entrada começa com um simples inteiro positivo numa linha, indicando o número de casos seguintes, cada um destes como descritos abaixo. Esta linha é seguida por uma linha em branco, e há também uma linha em branco entre duas entradas consecutivas.

Para cada labirinto de entrada, uma linha contém dois inteiros N e M para o número de linha e colunas do labirinto, respectivamente. Cada uma das N subseqüentes linhas contém M células de valores, separadas por espaços simples. As células de valores contêm inteiros maiores ou iguais a 1.

A SAÍDA

Para cada caso de teste, a saída deve seguir a descrição abaixo. As saídas de dois testes consecutivos são separadas por uma linha em branco.

Serão duas linhas: a primeira com as coordenadas, linha e coluna, da célula de partida, e a segunda com as coordenadas da célula de saída. Se houver uma série de soluções, imprima a com o menor ponto de partida lexicograficamente. Se permanecer um empate, imprima a com o menor ponto de saída lexicograficamente.

EXEMPLO DE ENTRADA

```
1

10 11
1 6 5 2 1 1 2 3 2 1 4
1 2 6 3 2 1 1 3 4 5 6
1 2 3 2 1 3 2 5 6 4 2
2 3 1 2 2 3 3 4 5 2 1
3 4 2 3 4 5 3 2 1 4 2
4 3 4 4 5 6 4 3 2 5 3
5 4 2 1 2 3 4 4 3 6 4
6 5 3 2 3 4 5 5 4 1 1
1 6 4 3 5 5 6 6 1 2 3
2 1 5 1 6 6 1 2 2 3 4
```

EXEMPLO DE SAÍDA

```
1 6
10 3
```

PROBLEMA E
DESTINO DOS FUNCIONÁRIOS

Este problema foi elaborado pela Prof.ª Dr.ª Maria Claudia Silva Boeres (DI/UFES - ES) e pela Prof.ª Dr.ª Maria Cristina Rangel (DI/UFES - ES). O problema é uma versão adaptada de um problema criado pela Prof.ª Clícia Valadares (DMAT/UERJ – RJ).

O presidente de uma empresa paulista de desenvolvimento de software pretende destinar cinco de seus melhores funcionários para coordenar projetos por dois anos em cinco filiais de sua empresa, existentes em cinco diferentes capitais no Brasil. Os nomes dos funcionários são: Ana Peixoto, Carlos Eduardo Novaes, Fábio Siqueira, Graziela de Alencar e Patrícia Pilar. As cidades são: Vitória, Salvador, Rio de Janeiro, Curitiba e Manaus. De acordo com a natureza dos projetos desenvolvidos em cada cidade, o presidente possui uma preferência de alocação funcionário/projeto, determinada pelo que ele julga ser o melhor perfil de profissional para coordenar cada projeto. Da mesma forma, cada funcionário possui uma preferência determinada pela sua identificação com a cidade que ele vai morar por dois anos.

O presidente da empresa precisa garantir que em cada cidade exista pelo menos um funcionário (do grupo selecionado por ele) coordenando um projeto. Por outro lado, cada funcionário pode se negar a se mudar para, no máximo, duas das cinco capitais candidatas a novas moradias, desde que sempre seja possível alocar um funcionário para cada filial da lista de capitais.

A tabela abaixo mostra a preferência do presidente. Ela é preenchida por números de 1 a 5 representando a escolha do presidente, onde o número 1 representa a primeira escolha, o número 2 a segunda, e assim por diante.

Pref. do Presidente	Vitória	Salvador	Rio de Janeiro	Curitiba	Manaus
Ana Peixoto	1	5	2	3	4
Carlos Eduardo Novaes	5	2	4	1	3
Fábio Siqueira	2	1	3	5	4
Graziela de Alencar	2	3	5	4	1
Patrícia Pilar	4	3	1	2	5

A próxima tabela mostra a preferência dos funcionários seguindo os mesmos critérios de numeração da tabela acima, ou seja, o número 1 representa a primeira escolha do funcionário, o número 2 a segunda, e assim por diante. Nesta tabela, o símbolo "-" representa o fato do funcionário se recusar a ir morar naquela cidade.

Pref. dos Funcionários	Vitória	Salvador	Rio de Janeiro	Curitiba	Manaus
Ana Peixoto	1	5	4	2	3
Carlos Eduardo Novaes	2	-	1	3	-
Fábio Siqueira	1	3	2	4	5
Graziela de Alencar	-	3	2	-	1
Patrícia Pilar	1	2	3	-	4

O problema consiste em encontrar o melhor emparelhamento perfeito que atenda às

restrições e concilie as preferências dos funcionários e do presidente. Um emparelhamento perfeito é definido pelo conjunto de associações entre funcionários/capitais de maneira que todos os funcionários sejam designados para todas as capitais, cada um associado a uma capital distinta.

A ENTRADA

Considere que a entrada sempre respeitará a ordem dos funcionários/capitais disposta nas tabelas apresentadas. Além disso, a definição de impossibilidade de moradia pelos funcionários não poderá inviabilizar a construção de uma solução, ou seja, estabelecer um emparelhamento perfeito. Uma forma de representar as informações das duas tabelas acima em uma só seria considerando o somatório das prioridades estabelecidas pelo presidente e pelos funcionários para cada capital, como mostra a tabela abaixo:

Pref. dos Funcionários	Vitória	Salvador	Rio de Janeiro	Curitiba	Manaus
Ana Peixoto	2	10	6	5	7
Carlos Eduardo Novaes	7	-	5	4	-
Fábio Siqueira	3	4	5	9	9
Graziela de Alencar	-	6	7	-	2
Patrícia Pilar	5	5	4	-	9

Assim, a entrada do problema consiste nos dados da tabela acima, ou seja, uma matriz 5x5, cuja primeira coluna contém os somatórios para Vitória, a segunda para Salvador, e na seqüência: Rio de Janeiro, Curitiba e Manaus. Já cada linha representa as informações sobre Ana, Carlos, Fábio, Graziela e Patrícia, exatamente nessa ordem. Cada célula da matriz é um inteiro ou um "-" e são separadas por espaços simples.

A entrada conterá um ou mais casos de testes. Uma linha em branco separa dois testes distintos. A entrada é terminada por um fim de arquivo.

A SAÍDA

Considerando as informações fornecidas nas tabelas acima, elabore um algoritmo que produza o melhor emparelhamento perfeito possível que concilie as preferências do presidente e dos funcionários e que respeite as restrições impostas ao problema (informações em negrito).

Dessa forma, a saída de cada teste consiste de cinco linhas contento o emparelhamento cidade funcionário, no seguinte formato:

Vitoria F_1
Salvador F_2
Rio de Janeiro F_3
Curitiba F_4
Manaus F_5, em que F_i é o primeiro do nome funcionário que vai para a cidade correspondente.

O resultado de dois testes diferentes será separado por uma linha em branco.

EXEMPLO DE ENTRADA

```
2 10 6 5 7
7 - 5 4 -
3 4 5 9 9
- 6 7 - 2
5 5 4 - 9
```

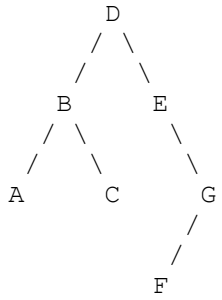
EXEMPLO DE SAÍDA

```
Vitoria Ana
Salvador Fabio
Rio de Janeiro Patricia
Curitiba Carlos
Manaus Graziela
```

PROBLEMA F
RECUPERAÇÃO DE ÁRVORE

A pequena Valentina gostava muito de brincar com árvores binárias. Seu favorito jogo era construir randomicamente árvores binárias com letras nos nós.

Este é um exemplo de uma das suas criações:



Para guardar suas árvores para as futuras gerações, ela escreveu duas strings para cada árvore: uma que a representa no formato pré-ordem (raiz, sub-arvore esquerda, sub-arvore direita) e outra que a representa no formato em ordem (sub-arvore esquerda, raiz, sub-arvore direita).

Para a árvore desenhada acima sua representação em pré-ordem é DBACEGF a em ordem é ABCDEFG.

Ela imaginava que tal par de strings daria informação suficiente para reconstruir a árvore mais tarde (mas ela nunca tentou isto).

Agora, anos mais tarde, olhando novamente para as strings, ela percebeu que reconstruir as árvores era realmente possível. Contudo, fazer a reconstrução na mão, será tedioso.

Então, agora ela pede para você escrever um programa que faça esse trabalho para ela!

A ENTRADA

A entrada conterá um ou mais casos de testes. Cada teste consiste de uma linha contendo duas strings preord e emord, representando o formato pré-ordem e em ordem da árvore binária, respectivamente. Ambas strings consistem unicamente de letras sem repetição (Assim elas não serão maiores que 26 caracteres).

A entrada é terminada por um fim de arquivo.

A SAÍDA

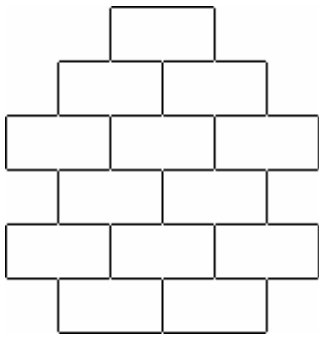
Para cada caso de teste, recupere a árvore binária da Valentine e imprima uma linha contendo a árvore no formato pós-ordem (sub-arvore esquerda, sub-arvore direita,raiz).

EXEMPLO DE ENTRADA
DBACEGF ABCDEFG ABC BAC
EXEMPLO DE SAÍDA
ACBFGED BCA

PROBLEMA G
CONSTRUINDO TORRES

Há N blocos numa caixa de brinquedos com 1-unidade de altura. O professor organiza um jogo de construir torres. A torre é construída por blocos. A altura da torre é H (h andares). A base da torre contém exatamente M blocos; e para cada andar acima, cada um deve conter um número de blocos que é exatamente um a menos ou a mais que o número de blocos do andar diretamente abaixo deste.

Abaixo, tem-se um exemplo de uma torre de H=6, M=2 e N=13.



Sua tarefa é determinar quantos tipos diferentes de torres podem ser obtidos. Duas torres são consideradas diferentes se há pelo menos um número i (1<i<=H) tal que o i-ésimo andar de uma torre contém um número diferente de blocos do i-ésimo andar de outra torre.

Cada torre é representada por um array de H inteiros positivos que determinam a estrutura da árvore, desde a base (elemento 1 do array) ao topo (elemento H do array). Cada um destes arrays são armazenados em forma ascendente, ou seja, 2 3 2 1 2 1 < 2 3 2 3 2 1, para os dados do exemplo anterior.

Dica: Você não precisa usar todos os blocos para construir um tipo de torre.

A ENTRADA

Para cada caso de teste:

- A primeira linha contém três números positivos N, H e M (N <= 32767, H<=60, M<=10).
- Cada uma das linhas seguintes (exceto a última) contém um inteiro positivo k. A última linha contém o número -1.

A entrada conterá um ou mais casos de testes. A entrada é terminada por um fim de arquivo.

A SAÍDA

Para cada caso de teste:

- A primeira linha contém o total de torres diferentes que podem ser obtidas.
- Cada uma das linhas seguintes contém um array que representa a estrutura da árvore correspondente ao número k da entrada.

Os resultados de duas entradas de teste são separados por uma linha em branco.

EXEMPLO DE ENTRADA

```
22 5 4
1
10
-1
```

EXEMPLO DE SAÍDA

```
10
4 3 2 1 2
4 5 4 5 4
```

PROBLEMA H
FIREBALL

O cientista Vitka Korneev criou e tem testado sua nova invenção: uma fireball. A fireball parece uma invenção geniosa! Sua principal característica é a capacidade de permanecer estável durante um número pré-definido de obstáculos. Esta característica é chamada de N-estabilidade: a fireball é N-estável se continua estável após N colisões, mas explode após a (N+1)ª colisão.

Então, você deve considerar que uma fireball N-estável perde um nível de estabilidade e torna-se (N-1)-estável após cada colisão com a parede. Por exemplo: uma fireball simples é 0-estável. Assim, com esta invenção torna-se possível atingir um inimigo com a fireball depois de uma série de rebatidas com as paredes. Ou seja, o valor militar desta invenção é inquestionável. Além disso, a fireball (N>0) te um comportamento um tanto incomum: ela ricocheteia somente de paredes de concreto! Assim, voa facilmente através de outros obstáculos. Este fato, como você pode pensar, resulta em um valor militar adicional à nova invenção: agora não é necessário providenciar uma trajetória limpa para lançar a fireball – ela voará através de qualquer obstáculo antes de atingir o inimigo.

Mas há um longo caminho para o primeiro protótipo ser feito. Primeiramente, é necessário investigar a trajetória que a fireball voa. O seguinte experimento foi preparado para este propósito: numa sala retangular dois pontos A e B são escolhidos aleatoriamente. Um cientista fica no ponto A e o alvo é colocado no ponto B. O cientista cria uma fireball N-estável enquanto seu assistente calcula a direção de lançamento com ajuda de um programa especial. A direção de lançamento é selecionada de tal forma que a fireball ricocheteia exatamente N vezes e atinge o alvo. Ao mesmo tempo, isto deve ser feito com a menor trajetória possível.

Então, você tem que escrever esse programa especial para o cálculo da direção. Os cientistas dizem que todas as fireballs ricocheteiam nas paredes de acordo com a lei: "ângulo de incidência igual ao ângulo de reflexão". Além disso, depois de uma colisão com uma das quatro quinas, ela ricocheteia exatamente na direção oposta. A teoria da fireball diz que uma colisão com uma quina é igual duas colisões com a parede. Assim, uma fireball 2-estável explode depois da segunda colisão se a primeira foi com uma quina da sala. Finalmente, assuma que a fireball é um ponto que se move em linha reta com velocidade constante.

A ENTRADA

A primeira linha contém dois números – a largura W e o comprimento H da sala. A segunda linha contém o número N (N-estabilidade da fireball). A terceira linha contém quatro números, que são as coordenadas dos pontos A e B.

Todos os números são inteiros e são separados por um espaço simples. Os pontos A e B estão dentro da sala, mas não na margem. A largura e o comprimento da sala não podem

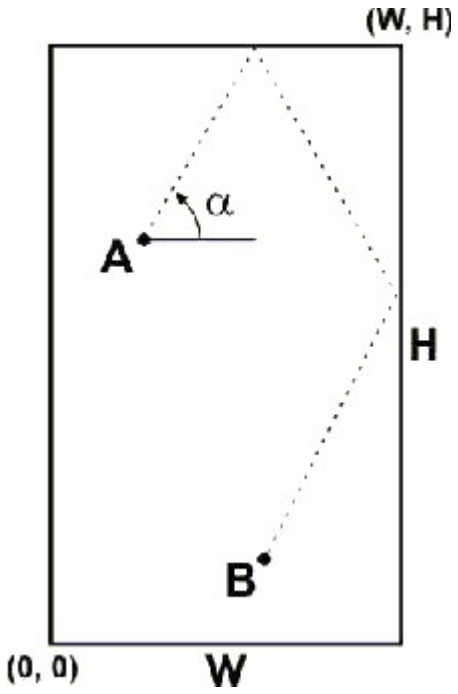
exceder 1000 e são maiores que 1. N está entre 0 e 100 inclusive.

A entrada conterá um ou mais casos de testes sem linhas em branco entre eles. A entrada é terminada por um fim de arquivo.

A SAÍDA

Para cada teste, imprima uma linha com o ângulo em graus (com 2 dígitos após o ponto decimal), que dá a desejada direção da fireball. Se há uma série de ângulos, seu programa deve imprimir o menor deles.

O ângulo e as coordenadas são medidos como mostra a figura abaixo.



EXEMPLO DE ENTRADA

```
1000 10
3
101 5 128 8
```

EXEMPLO DE SAÍDA

```
45.00
```

PROBLEMA I
MATRIZES ESPARSAS

Este questão é uma versão adaptada, pela comissão organizadora, do problema proposto e elaborado pela Prof.ª Dr.ª Lucia Catabriga (DI/UFES – ES).

Sistemas lineares $Ax=b$ são uns dos modelos matemáticos mais utilizados na solução de problemas reais das mais diversas áreas do conhecimento. A matriz dos coeficientes A de ordem n x n e o vetor dos termos independentes b de ordem n são conhecidos e o vetor x de ordem n é a solução. Em geral a matriz dos coeficientes possui uma quantidade expressiva de coeficientes nulos sendo denominada **Matriz Esparsa**. Os algoritmos numéricos de solução são tão mais eficientes quanto mais otimizada for a forma de armazenar a matriz A, pois economizam memória, e mais importante, realizam um número menor de operações. Existem duas estratégias de solução aproximada: métodos diretos e métodos iterativos. Quando os sistemas são de grande porte (ordem de grandeza de milhões de incógnitas) o método iterativo é a opção mais vantajosa. Uma das operações mais caras computacionalmente no processo de solução de sistemas lineares por métodos iterativos é o produto matriz vetor.

O método de armazenamento denominado (CSR - *Compressed sparse Row*) consiste em armazenar somente os coeficientes não nulos em um vetor linha por linha. Considere que uma matriz de ordem n x n tenha nnz coeficientes não nulos (nnz < n^2). Para tal é necessário definir 3 vetores:

- AA: vetor para armazenar linha por linha somente os coeficientes não nulos com nnz posições,
- JA: um vetor contendo o índice da coluna dos elementos de AA com nnz posições,
- IA: um vetor contendo os índices de AA onde inicializam cada linha. Para facilitar um conjunto de operações considera-se esse vetor com n+1 posições sendo que a (n+1)-ésima posição contem o valor (nnz+1).

$$A=\begin{bmatrix} k_{11} & k_{12} & & k_{14} & & & & & \\ k_{21} & k_{22} & k_{23} & k_{24} & k_{25} & & & & \\ & k_{32} & k_{33} & & k_{35} & k_{36} & & & \\ k_{41} & k_{42} & & k_{44} & k_{45} & & k_{47} & & \\ & k_{52} & k_{53} & k_{54} & k_{55} & k_{56} & & k_{58} & k_{59} \\ & & k_{63} & & k_{65} & k_{66} & & & k_{69} \\ & & & k_{74} & & & k_{77} & k_{78} & \\ & & & k_{84} & & & k_{87} & k_{88} & k_{89} \\ & & & & k_{95} & k_{96} & & k_{98} & k_{99} \end{bmatrix}$$

$$AA=\begin{bmatrix} \overset{1}{\frown} & & & \overset{4}{\frown} & & & & \overset{9}{\frown} & & & \overset{36}{\frown} & & & \\ k_{11} & k_{12} & k_{14} & k_{21} & k_{22} & \dots & k_{95} & k_{96} & k_{33} & k_{98} & k_{99} & k_{95} & k_{96} & k_{98} & k_{99} \end{bmatrix}$$

$$JA=[1 \quad 2 \quad 4 \quad 1 \quad 2 \quad \dots \quad 5 \quad 2 \quad 3 \quad \dots \quad 9 \quad 5 \quad 6 \quad 8 \quad 9]$$

$$IA=[1 \quad 4 \quad 9 \quad 13 \quad 18 \quad 25 \quad 29 \quad 32 \quad 36 \quad 40]$$

Para tal tipo de armazenamento, escreva um código que implemente o produto matriz vetor e calcule para cada caso o número de operações de ponto flutuante (adição/subtração, multiplicação/divisão) para realizar o produto matriz vetor. Seu código deve receber a matriz esparsa e um vetor **v**, gerar a matriz otimizada para o armazenamento CSR, calcular o produto matriz vetor (A*v) e contar o número de operações de ponto flutuante realizadas durante o produto.

A ENTRADA

A primeira linha contém o número de testes que serão realizados. A próxima linha está em branco. Para cada teste, tem-se:

- Uma linha contendo a dimensão n na matriz A (n <= 15)
- Cada uma das outras n linhas contém n inteiros: os elementos de cada linha da matriz A.
- Uma linha contendo os elementos do vetor **v**, um vetor de n elementos, e que será multiplicado pela matriz de coeficientes.

Obs.: Os elementos do vetor e da matriz são separados por um espaço simples. Antes dos dados de um novo teste acrescenta-se uma linha em branco.

A SAÍDA

Para cada teste o resultado é composto de 5 linhas:

- Em cada uma das três primeiras, imprimem-se os elementos do vetor AA, JA e IA.
- Na próxima linha, será impresso o vetor contendo o resultado da multiplicação A*v.
- Na última linha deve estar o total de operações de ponto flutuante efetuadas na operação A*v, separadas por um espaço simples.

Obs.: Os elementos dos vetores são separados por um espaço simples. Antes dos resultados de um novo teste acrescenta-se uma linha em branco.

EXEMPLO DE ENTRADA

```
1

7
7 1 0 0 2 0 3
0 1 8 1 0 2 0
2 0 1 0 7 0 2
1 0 0 9 1 0 0
0 2 1 0 1 4 0
0 1 1 0 0 3 1
2 0 1 0 0 2 5
1 1 1 1 1 1 1
```

EXEMPLO DE SAÍDA

```
7 1 2 3 1 8 1 2 2 1 7 2 1 9 1 2 1 1 4 1 1 3 1 2 1 2 5
1 2 5 7 2 3 4 6 1 3 5 7 1 4 5 2 3 5 6 2 3 6 7 1 3 6 7
1 5 9 13 16 20 24 28
13 12 12 11 8 6 10
47
```