Banda

Nome do arquivo fonte: `banda.c`, `banda.cpp`, *ou* `banda.pas`

Jimmy é um garoto muito esperto que adora música. No último mês ele ganhou um campeonato de um jogo cujo objetivo é tocar guitarra. Empolgado, Jimmy decidiu montar uma banda. Para Jimmy a banda perfeita tem quatro integrantes, ele e mais três: um baterista, um baixista e um cantor.

Agora Jimmy precisa encontrar os outros integrantes da banda. Para isto ele reuniu todos os álbums que encontrou na internet e, após escutá-los diversas vezes, compilou o que ele chama de *lista de entrosamento entre músicos*. Nessa lista ele atribui, para cada par de músicos que já tocaram juntos, uma nota inteira de 1 a 100, que é uma medida de quão bem os músicos tocam juntos (o *nível de entrosamento* entre eles). Se dois músicos nunca tocaram juntos o nível de entrosamento é zero. Jimmy nunca tocou com nenhum músico da lista.

Jimmy pretende formar a sua banda a partir da lista de entrosamento entre músicos, da seguinte maneira: ele quer escolher os outros três músicos de tal forma que a soma dos níveis de entrosamento dos integrantes da banda seja a maior possível (ou seja, a soma dos níveis de entrosamento dos três pares possíveis de serem formados entre os três novos integrantes seja a maior possível).

Mas a lista de entrosamento entre músicos ficou muito grande e Jimmy não está conseguindo escolher os integrantes. Por isso, Jimmy está pedindo sua ajuda.

Tarefa

Você deve ajudar Jimmy a montar a melhor banda possível fazendo um programa que receba uma lista contendo o nível de entrosamento para cada par de músicos que já tocaram junto, e determine os músicos que formariam a melhor banda.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado).

A primeira linha da entrada é formada por dois inteiros N e M, informando respectivamente o número de músicos ($3 \leq N \leq 100$) e o número de pares de músicos que já tocaram juntos ($0 \leq M \leq 10^4$). Os músicos são identificados por números inteiros de 1 a N. Cada uma das M linhas seguintes contém três inteiros X, Y e Z, em que X e Y representa um par de músicos ($1 \leq X \leq N$, $1 \leq Y \leq N$ e $X \neq Y$) e Z representa o seu nível de entrosamento ($1 \leq Z \leq 100$). Cada par de músicos que já tocou junto aparece uma única vez na entrada.

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha, contendo três números inteiros separados por espaço em branco, identificando os três outros músicos que devem compor a banda (em qualquer ordem). Se existir mais de uma melhor banda, Jimmy contenta-se com qualquer uma.

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 30 pontos, $N \leq 10$ e $M \leq 100$.
- Em um conjunto de casos de teste que totaliza 80 pontos, $N \leq 50$ e $M \leq 2450$.

Exemplos

Entrada	**Saída**
3 3 1 2 50 2 3 27 3 1 1	1 2 3

Entrada	**Saída**
5 8 1 2 50 1 3 50 1 4 50 2 3 50 2 5 10 3 4 50 3 5 25 4 5 20	1 3 4

Ortografia

Nome do arquivo fonte: `ortografia.c`, `ortografia.cpp` *ou* `ortografia.pas`

Um serviço de busca na Internet está preocupado com a crescente taxa de erros de ortografia de seus usuários, tornando mais difíceis as buscas por palavras chaves, que constantemente contêm erros de algumas letras, devidos a má digitação ou má ortografia.

O serviço funciona com base num dicionário de palavras. O usuário deve inserir uma palavra num campo de um formulário; o serviço então procura esta palavra no dicionário e retorna conteúdo que tenha relação com a palavra.

Para contornar o problema de ortografia, você foi contratado para fazer um programa que tenta adivinhar qual palavra o usuário pretendia procurar, independente de haver erros de ortografia nela.

Para este problema vamos definir a *distância* entre duas palavras A e B como sendo o número de operações, descritas abaixo, necessárias para transformar A em B:

1. Retirar uma letra de A.
2. Adicionar uma letra a A, em qualquer posição.
3. Trocar qualquer letra de A por outra letra, na mesma posição.

O serviço de busca definiu que a palavra P fornecida pelo usuário pode se referir a uma palavra D do dicionário se está a uma distância de no máximo 2 de D.

Exemplos:

- A palavra 'tu' pode se referir à palavra do dicionário 'tubo', realizando duas vezes a operação 2.
- A palavra 'crto' pode se referir à palavra do dicionário 'corte', realizando uma vez a operação 2 e uma vez a operação 3.
- A palavra 'crto' pode se referir à palavra do dicionário 'curto', realizando uma vez a operação 2.
- A palavra 'hortgrafea' não pode se referir à palavra do dicionário 'ortografia'.

Você deve escrever um programa que, dado um dicionário de palavras, descubra para cada palavra fornecida pelo usuário a quais palavras do dicionário ela pode se referir, nas condições descritas acima.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado). A primeira linha contém 2 inteiros N, M, representando respectivamente o número de palavras contidas no dicionário ($1 \leq N \leq 1000$) e o número de palavras a serem analisadas ($1 \leq M \leq 100$). Cada uma das N linhas seguintes conterá uma palavra pertencente ao dicionário. Cada uma das M linhas seguintes conterá uma palavra a ser analisada, fornecida pelo usuário. Cada palavra pode ter de 1 a 20 letras, contendo apenas letras de 'a' a 'z', minúsculas.

Saída

Seu programa deve imprimir, na *saída padrão*, M linhas, sendo uma linha para cada palavra fornecido pelo usuário. Cada linha deve conter todas palavras do dicionário às quais a palavra fornecida pode se referir. No caso de haver mais de uma palavra em uma linha da resposta, elas devem ser separadas por um espaço em branco, apareçendo na ordem que elas foram dadas na entrada, como pode ser visto no exemplo de saída abaixo. No caso de não haver nenhuma palavra em uma linha da resposta, deixe-a em branco.

Exemplo de entrada	**Exemplo de saída**
3 3 pato pateta caneca pat ccanecos pata	pato pato pateta

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 30 pontos, $N \leq 10$ e $M \leq 10$.
- Em um conjunto de casos de teste que totaliza 55 pontos, $N \leq 50$ e $M \leq 500$.

Pedido de Desculpas

Arquivo fonte: desculpa.c, desculpa.cc, desculpa.cpp ou desculpa.pas

Cuca saiu para jogar futebol com os amigos e esqueceu do encontro que tinha com a namorada. Ciente da mancada, Cuca deseja elaborar um pedido especial de desculpas. Resolveu então enviar flores e usar o cartão da floricultura para escrever um pedido especial de desculpas.

Cuca buscou na internet um conjunto de frases bonitas contendo a palavra 'desculpe' (que pode ocorrer mais de uma vez na mesma frase). No entanto, o cartão da floricultura é pequeno, e nem todas as frases que Cuca colecionou poderão ser aproveitadas.

Cuca quer aproveitar o espaço do cartão, onde cabe um número limitado de caracteres, para escrever um sub-conjunto das frases coletadas de modo que apareça o máximo de vezes possível a palavra 'desculpe'.

Tarefa

Escreva um programa que, dados o número de caracteres que cabem no cartão e a quantidade de frases coletadas (com os respectivos comprimentos e os números de ocorrências da palavra 'desculpe'), determine o número máximo de vezes que a palavra aparece, utilizando apenas as frases colecionadas, sem repetí-las.

Entrada

A entrada é constituída de vários casos de teste. A primeira linha de um caso de teste contém dois números inteiros C e F indicando respectivamente o comprimento do cartão em caracteres ($8 \leq C \leq 1000$) e o número de frases coletadas ($1 \leq F \leq 50$). Cada uma das F linhas seguintes descreve uma frase coletada. A descrição é composta por dois inteiros N e D que indicam respectivamente o número de caracteres na frase ($8 \leq N \leq 200$) e quantas vezes a palavra 'desculpe' ocorre na frase ($1 \leq D \leq 25$). O final da entrada é indicado por $C = F = 0$.

A entrada deve ser lida do dispositivo de entrada padrão (normalmente o teclado).

Saída

Para cada caso de teste seu programa deve produzir três linhas na saída. A primeira identifica o conjunto de teste no formato "`Teste n`", onde `n` é numerado a partir de 1. A segunda linha deve conter o máximo número de vezes que a palavra 'desculpe' pode aparecer no cartão, considerando que apenas frases coletadas podem ser utilizadas, e cada frase não é utilizada mais de uma vez. A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

A saída deve ser escrita no dispositivo de saída padrão (normalmente a tela).

Restrições

$8 \leq C \leq 1000$ ($C = 0$ apenas para indicar o fim da entrada)
$1 \leq F \leq 50$ ($S = 0$ apenas para indicar o fim da entrada)
$8 \leq N \leq 200$
$1 \leq D \leq 25$

Exemplo de Entrada	**Saída para o Exemplo de Entrada**
200 4 100 4 100 1 120 2 80 5 40 3 10 1 10 1 20 2 0 0	Teste 1 9 Teste 2 4