Problema A

Extenso

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

Autor: Antonio Cesar de Barros Munari (Fatec Sorocaba)

Chiquinho foi contratado para trabalhar no setor de Informática de uma pequena empresa de sua cidade. Uma de suas primeiras tarefas é melhorar a qualidade de alguns serviços de preenchimento de documentos utilizados pelo pessoal do setor administrativo. Um deles requer a geração da versão por extenso de valores monetários, requeridos por alguns tipos de contratos e, em menor escala, no preenchimento de cheques e notas promissórias. Você, amigo de Chiquinho, foi acionado por ele para produzir um programa capaz de atender a esse propósito.

Entrada

Existem vários casos de teste. Cada um é composto por uma única linha contendo um valor V ($0.00 < V \leq 1000000.00$), que corresponde a uma quantia em dinheiro, expressa em reais, para a qual deve ser gerada a correspondente versão por extenso. A entrada é finalizada com um valor V igual a zero, que não deverá ser processado.

Saída

Para cada caso de teste, imprima uma linha contendo uma *string* em maiúsculas com o valor em extenso do número lido na entrada, conforme os exemplos apresentados. Não utilizar acentos nem vírgulas ao imprimir os resultados.

Exemplo de entrada

```
10.90
1.00
1.01
2.50
1000.00
1000000.00
837052.20
0.35
0.00
```

Exemplo de saída

```
DEZ REAIS E NOVENTA CENTAVOS
UM REAL
UM REAL E UM CENTAVO
DOIS REAIS E CINQUENTA CENTAVOS
UM MIL REAIS
UM MILHAO DE REAIS
OITOCENTOS E TRINTA E SETE MIL E CINQUENTA E DOIS REAIS E VINTE CENTAVOS
TRINTA E CINCO CENTAVOS
```

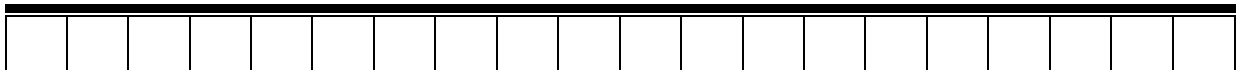
Problema B

Estacionamento

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

Autor: Leandro Luque (Fatec Mogi das Cruzes)

O estacionamento dos professores da Fatec tem várias vagas perpendiculares à parede que acompanha toda sua extensão, conforme figura seguinte.



Por estar localizado na frente de uma avenida muito movimentada, os professores não tem tempo para escolher vagas. Na primeira vaga disponível que encontra, da esquerda para a direita, o professor estaciona o seu carro. Caso não existam vagas disponíveis, o professor estaciona o carro em uma rua próxima à faculdade.

O diretor da Fatec deseja avaliar quantos professores estão estacionando seus carros na rua a partir de informações do horário de chegada e saída de cada professor. Escreva um programa para ajudá-lo. Assuma que um professor que chega no mesmo momento que outro está saindo não consegue estacionar no lugar que está sendo liberado.

Entrada

A entrada é composta por diversos casos de teste. A primeira linha de cada caso de teste contém dois números inteiros V ($1 \leq V \leq 50$) e P ($1 \leq P \leq 100$), que representam, respectivamente, o número de vagas do estacionamento e o número de professores da Fatec. As P linhas seguintes contêm quatro números inteiros HE ($0 \leq HE \leq 23$), ME ($0 \leq ME \leq 60$), HS ($0 \leq HS \leq 23$) e MS ($0 \leq MS \leq 60$), que representam a hora de entrada, minuto de entrada, hora de saída e minuto de saída de cada professor, respectivamente. Assuma que um professor nunca fica mais que 24 horas na Fatec e, após sair, também não volta nas próximas 24 horas. A última linha da entrada é composta por "0 0".

Saída

Para cada caso de teste, imprima uma linha contendo o número de professores que tiveram que estacionar seu carro na rua próxima à Fatec.

Exemplo de entrada

```
5 8
12 50 12 59
12 51 14 00
12 51 14 30
12 52 14 00
12 55 20 00
12 56 15 00
12 58 20 00
13 00 15 00
0 0
```

Exemplo de saída

```
2
```

Problema C

Polinômios

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

Autor: Antonio Cesar de Barros Munari (Fatec Sorocaba)

O professor de Matemática está muito preocupado com as dificuldades que os seus alunos estão apresentando em sua matéria, mesmo em temas simples como a manipulação de polinômios. Para que os estudantes consigam melhorar o aproveitamento, ele aumentou a quantidade de listas de exercícios e também conseguiu a contratação de um monitor para o auxiliar nas atividades de reforço da disciplina. O felizardo escolhido para monitor é você, que já demonstrou incríveis habilidades matemáticas e impressionantes dotes em programação de computadores. Como primeira tarefa, você foi encarregado de criar um programa capaz de avaliar polinômios.

Entrada

A entrada possui vários casos de teste. Inicialmente será lido um inteiro positivo $N \leq 100$, que indica a quantidade de casos de teste a serem processados. Cada caso de teste é iniciado com uma linha contendo o polinômio a ser avaliado. Assuma que teremos apenas polinômios válidos de uma única variável (no caso será a variável *x*), cada um deles sendo expresso por meio de uma *string* de até 30 caracteres composta apenas pelos sinais '+', '-', a letra 'x' minúscula e dígitos numéricos entre 0 e 9. Por exemplo, o polinômio $3x^5 + x^3 - 2x - 1$ seria representado na entrada como `3x5+x3-2x-1`. Em seguida um inteiro V ($0 < V \leq 10$) indica a quantidade de valores a serem considerados para a variável do polinômio. Segue então uma linha contendo V inteiros X ($-9 \leq X \leq 9$) separados por espaço em branco, que são os valores a serem considerados como o conteúdo da variável `x` do polinômio.

Saída

Para cada caso de teste, você deve imprimir uma linha contendo o seu número e os resultados da avaliação do polinômio para cada possível valor de `x`, conforme os exemplos apresentados.

Exemplo de entrada

```
2
3x5+x3-2x-1
4
0 1 2 3
-3x2+5x3+x-25
3
-1 3 8
```

Exemplo de saída

```
Caso 1: -1 1 99 749
Caso 2: -34 86 2351
```

Problema D

Regra dos Terços

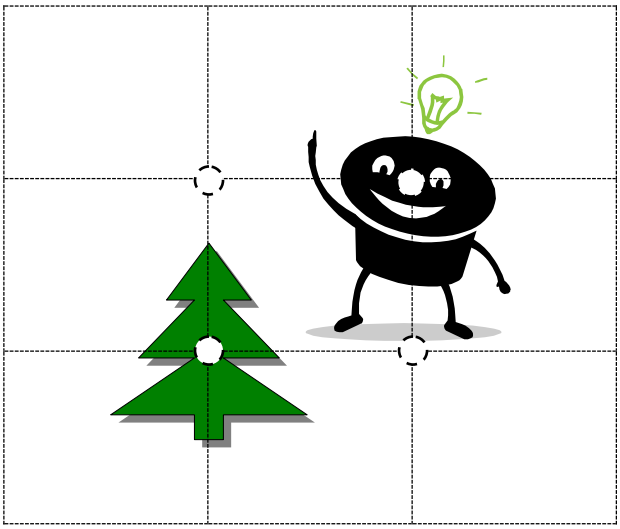
Nome do arquivo fonte: `terco.c`, `terco.java`, `terco.cpp` *ou* `terco.pas`

Autor: Leandro Luque (Fatec Mogi das Cruzes)

Em uma temporada fora do país, Jefferson matriculou-se no curso de fotografia "Foto Para Los Primos". Baseando-se no conteúdo estudado no curso, ele tirou fotos muito bonitas da viagem e deu dicas para seus amigos quanto a algumas técnicas aprendidas.

Uma das técnicas ensinadas por Jefferson a seus amigos foi a "Regra dos Terços". Segundo esta regra, o objeto de interesse em uma foto deve estar posicionado em uma das intersecções formadas pela divisão da foto em 9 quadros de igual tamanho, traçando 2 linhas verticais e horizontais imaginárias.

A imagem seguinte ilustra a aplicação desta regra. Os pontos representam os locais onde devem estar posicionados os objetos de interesse. Pode-se perceber que, nesta imagem, tanto o boneco quanto a árvore estão posicionados sobre alguma intersecção.



Para garantir que as fotos dos amigos também ficarão boas, Jefferson decidiu escrever um programa para câmeras digitais que especificará se um objeto de interesse está posicionado de acordo com a Regra dos Terços.

Entrada

A entrada é composta por vários casos de teste. A primeira linha de cada caso de teste contém quatro números reais com duas casas decimais cada, X_A, Y_A, X_B, Y_B ($0 \leq X_A < 100$, $0 < Y_A \leq 100$, $X_A < X_B \leq 100$, $0 \leq Y_B < Y_A$), separados por um espaço, que representam as coordenadas no plano cartesiano do canto superior esquerdo (X_A,Y_A) e do canto inferior direito (X_B, Y_B) da cena fotografada. A linha seguinte contém quatro números reais com duas casas decimais cada, X_C, Y_C, X_D, Y_D ($X_A \leq X_C < X_B$, $Y_B < Y_C \leq Y_A$, $X_C < X_D \leq X_B$, $Y_B \leq Y_D < Y_C$), separados por um espaço, que representam as coordenadas no plano cartesiano do canto superior esquerdo (X_C,Y_C) e do canto inferior direito (X_D, Y_D) do retângulo que engloba o objeto de interesse. Após o último caso de teste, é especificada uma linha contendo "0.00 0.00 0.00 0.00".

Saída

Para cada caso de teste, imprima uma linha contendo o caractere maiúsculo ‘V’ se o retângulo do objeto de interesse estiver localizado sob algum dos pontos de intersecção da regra dos terços. Caso contrário, imprima o caractere maiúsculo ‘F’.

Exemplo de entrada

```
0.00 9.00 9.00 0.00
2.00 7.00 3.00 4.00
0.00 9.00 9.00 0.00
4.00 5.00 5.00 1.00
0.00 0.00 0.00 0.00
```

Saída para o exemplo de entrada

```
V
F
```

Problema E

Um perfeito capicua

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

Autor: Antonio Cesar de Barros Munari (Fatec Sorocaba)

A Teoria dos Números é o nome dado ao ramo da Matemática dedicado ao estudo de propriedades apresentadas pelos números. É uma área repleta de curiosidades, em que inúmeros termos exóticos são utilizados para descrever determinadas características numéricas. Este problema vai explorar dois desses termos.

Um número inteiro é chamado de 'capicua' caso ele seja igual ao reverso dele próprio. O seu primeiro dígito é igual ao seu último dígito, seu segundo dígito é igual ao seu penúltimo dígito, etc. Por exemplo, os inteiros 32423, 121, 55 e 123321 são capicuas. É o análogo numérico do conceito de palíndromo para as palavras.

Chamamos de 'quadrado perfeito' ao inteiro positivo cuja raiz quadrada é um inteiro positivo. São exemplos de quadrados perfeito os números 9, cuja raiz quadrada é 3, e 25, cuja raiz quadrada é 5.

Neste programa você deverá determinar quantos números existem em um intervalo que são capicuas e quadrados perfeitos.

Entrada

A entrada possui vários casos de teste, cada um composto por um inteiro N ($0 \leq N < 2000000$) que deverá ser verificado se é capicua e quadrado perfeito. Encerrar o processamento quando for lida uma entrada com $N = 0$.

Saída

Para cada caso de teste imprima uma linha contendo um 'S' maiúsculo e sem aspas se o inteiro correspondente for capicua e quadrado perfeito e 'N' maiúsculo e sem aspas em caso contrário.

Exemplo de entrada

```
12321
989
32423
123
121
0
```

Exemplo de saída

```
S
N
N
N
S
```

Problema F

Árvore binária ordenada

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

Autor: Antonio Cesar de Barros Munari (Fatec Sorocaba)

Uma estrutura de dados muito interessante para a representação de valores ordenados é a árvore, bastante estudada nos cursos da área de Computação. Árvores possuem uma tipologia bem variada, e as árvores binárias são particularmente populares. Uma árvore binária é aquela em que cada nó possui no máximo dois nós descendentes. Se as subárvores esquerda e direita de cada nó de uma árvore tiverem sempre uma quantidade muito parecida de elementos, podemos dizer que essa árvore é balanceada. Obviamente, a expressão 'quantidade muito parecida' é bastante vaga, e para fins práticos, precisamos especificar um critério que defina o seria aceito como 'muito parecido'. Um critério bastante comum é considerar como balanceada a árvore binária em que, para cada nó, a subárvore que tiver mais elementos tem, no máximo, um elemento a mais que a subárvore restante. Uma árvore binária é considerada ordenada se, para cada nó, todos os elementos de uma de suas subárvores são menores ou iguais a ele e todos os elementos da outra subárvore são maiores que ele.

Neste problema, você deverá analisar uma árvore binária ordenada balanceada em que os nós armazenam valores inteiros e determinar sua altura, sua quantidade de folhas e determinar o nível de um determinado valor dentro dessa árvore. Lembramos que uma folha é o nó que não possui descendentes; que a altura de uma árvore é a quantidade de ramos que existe entre a sua raiz e a sua folha mais distante. O nível de um nó é zero se esse nó é a raiz, é um se o nó é descendente direto da raiz, etc.

A título de exemplo, para os valores 3, 9, 8, 10, 11, 7, 1, 0, -1, 4, 5, 6, 13 e 15, teremos a árvore binária ordenada apresentada na figura 1. A altura dessa árvore é 3, ela possui 7 folhas e o nível do nó contendo o valor 0 é 3 e o nível do nó contendo o valor 8, por exemplo, é 2.

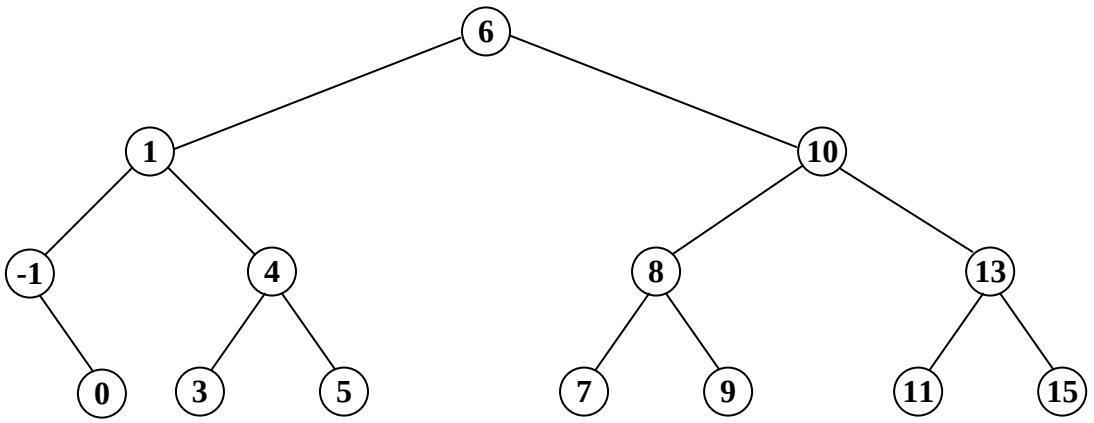


Figura 1: Árvore binária ordenada balanceada.

Entrada

O arquivo de entradas inicia com um inteiro *N* que indica a quantidade de casos de teste. Cada caso de teste é composto por duas linhas. A primeira inicia com um inteiro *Q* ($0 < Q \leq 32600$) que indica quantos nós terá a árvore, e seguem-se então *Q* inteiros *V* ($-32768 \leq V \leq 32767$), correspondendo aos valores de cada nó. A segunda linha contém um inteiro *P*, que é o valor para o qual o programa deve determinar o seu nível na árvore binária ordenada. Você pode

assumir que não existirão valores repetidos para serem armazenados na árvore e que o valor *P* a ser pesquisado sempre vai existir na árvore.

Saída

Para cada caso de teste, o programa deverá imprimir o seu número, conforme mostrado nos exemplos e, na linha seguinte, três inteiros que representam, respectivamente, a altura da árvore binária correspondente, a quantidade de folhas dessa árvore e o nível em que se encontra o valor *P* informado na entrada. Uma linha em branco deve ser impressa após cada caso de teste.

Exemplo de entrada

```
4
15 3 9 8 10 11 7 1 0 -1 4 5 6 13 15 16
0
7 7 6 5 4 2 1 0
4
11 4 -10 -8 28 36 42 56 68 9 3 1
36
14 3 9 8 10 11 7 1 0 -1 4 5 6 13 15
10
```

Exemplo de saída

```
Caso 1:
3 8 2

Caso 2:
2 4 2

Caso 3:
3 4 3

Caso 4:
3 7 1
```

Problem G

Hangman

Source file name: `hang.c`, `hang.java`, `hang.cpp` *or* `hang.pas`

Authors: Antonio Cesar de Barros Munari, Paulo Edson (Fatec Sorocaba)

Hangman is a traditional pastime in which a participant is challenged to guess a word or a secret phrase proposed by another person. In the version of the pastime in this problem, sentences are not allowed, i.e., in each round only a single word made up to 30 uppercase alphabetic characters must be guessed. The proposer of the riddle indicates the size of the word marking each letter of it with a single underline, beside the gallows. The contestant should give guesses of letters for several rounds. For each right guessing, the proposer should write the letter on the right underline. On the other hand, for each wrong guessing, the proposer draws a part of the human body on the gallows.

The challenge ends when the secret word is fully guessed by the contestant. The hanging takes place when the amount of errors exceeds the number of parts of the human body. For example, if the puppet is drawn with head, torso, two arms and two legs, there will be six parts and it will be hanged after the contestant's seventh wrong guessing. However, if the puppet is drawn head, left eye, right eye, nose, mouth, trunk, two arms and two legs, the hanging will take place only after the eleventh wrong guessing The amount of guessing rounds has to be decided by the participants of the challenge.

Johnny knows this game very well and wants to do an experiment, based on a class he had at school. He always wants to test how often the letters are used. For this he takes a text and counts the number of times each letter appears on it, and based on his assessment, tries to guess a secret word proposed by someone, following a descending order of the frequency the letters appear. If there are letters with the same number of occurrences, he decides based on the alphabetical order of them.

In this problem you will assist Johnny to simulate the operation of this strategy. An initial text will be provided to determine the frequency of the letters appearances and then several secret words will be presented. Your program should determine whether the strategy will make Johnny win or lose in each case.

Input

The input file begins with an integer C ($0 < C \leq 100$) indicating the amount of test cases. Each test case starts with a text with up to 10,000 characters, which serve as a basis for determining the frequency of each letter. Consider that there will neither be cedilla (ç) nor accented words. Then D is reported an integer ($0 < D \leq 100$) indicating the amount of challenges to be solved. There will be D lines, each of them consisting of one word with up to 30 uppercase alphabetic characters, which is the secret word to be guessed, and an integer P ($0 < P \leq 15$), which indicates the amount of parts that the puppet to be hanged will have when completed.

Output

For each test case, print the number as shown in the examples and then, for each challenge posed in this case, print a line with the character 'Y' (uppercase, no quotes) if the secret word can be completed when the letters are informed according to the criteria established by Johnny and 'N' (uppercase, no quotes) if he is hanged. Leave a blank line between two different test cases.

Input example

```
2
White man came across the sea Brought us pain and misery Killed our tribes killed our creed Took our game for his own need We fought him hard we fought him well Out on the plains we gave him hell But many came too much for Cree Oh will we ever be set free? Riding through dustclouds and barren wastes Galloping hard on the plains Chasing the redskins back to their holes Fighting them at their own game Murder for freedom the stab in the back Women and children and cowards attack Run to the hills run for your lives
5
INTERNET 5
PERPENDICULAR 7
PROFESSOR 15
AMERICA 5
SOFTWARE 9
I know you You always rode so tall in the saddle You are the answer But I know that you are only causin' trouble Take all you want, leave me the rest behind You're a user A winnin' no new friends, just make one small step You're a user You never came to anyone's call Spent all your life head on the wall While you're up there, there's only one way to fall All the king's horses and all the king's men Will never put you together again
3
FATEC 5
HARDWARE 10
PERPENDICULAR 12
```

Output example

```
Case 1:
Y
N
Y
N
Y

Case 2:
N
Y
Y
```

Problema H

Códigos das turmas

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

Autor: Antonio Cesar de Barros Munari (Fatec Sorocaba)

A secretaria da Fatec está precisando de um programa que determine as turmas que uma determinada disciplina precisará formar para atender à quantidade de alunos nela matriculados. Para cada caso, o programa deverá receber a sigla da disciplina, composta por duas letras, o código numérico do curso em que a disciplina será oferecida, formado por dois dígitos, um número definindo a quantidade máxima de alunos permitida em cada turma, seguida de 3 outros números, indicando respectivamente a quantidade de alunos inscritos para a disciplina nos períodos da manhã, da tarde e da noite. O programa deverá então exibir os códigos das turmas que precisarão ser formadas para atender a demanda, observadas as restrições de quantidade máxima de alunos por turma e do período letivo de cada aluno (não se pode matricular alunos da manhã em turmas da tarde, por exemplo). A fórmula para determinar a composição do código da turma é a seguinte:

- O código da turma é composto por 6 caracteres, no formato XX9X99.
- Os dois primeiros caracteres devem ser alfabéticos, em maiúsculas, e indicam a sigla da disciplina.
- O terceiro caracter é um número de 1 a 3 que indica o período letivo da turma: 1-manhã, 2-tarde, 3-noite.
- O quarto caracter é uma letra maiúscula que funciona como um identificador auxiliar da turma dentro do período letivo:
    - A-I, para turmas do período da manhã, sendo A para a primeira turma desse período, B para a segunda turma, etc;
    - J-O, para turmas do período da tarde, sendo J para a primeira turma desse período, K para a segunda turma, etc;
    - P-Z, para turmas do período da noite, sendo P a primeira turma desse período, Q a segunda turma, etc.
- Os dois últimos caracteres representam o código do campus onde a turma será montada.

Assim, por exemplo, o código ED2J07 indica que trata-se de uma turma da disciplina cuja sigla é ED, oferecida no período da tarde (essa é a primeira turma dessa disciplina nesse período) e que é oferecida no curso de código 07.

Entrada

A entrada se inicia com um inteiro N ($0 < N \leq 1000$) indicando a quantidade de casos de teste a serem processados. Cada caso é composto por uma única linha contendo uma *string* de dois caracteres alfabéticos D, que é a sigla da disciplina, um inteiro positivo P de dois dígitos, com um eventual zero a esquerda, indicando o prédio, um inteiro T ($0 < T \leq 100$) que representa a quantidade máxima de alunos permitida em uma turma daquela disciplina e três inteiros P_1, P_2 e P_3 ($0 < P_1, P_2, P_3 \leq 1000$) que indicam, respectivamente, a quantidade de alunos inscritos para cursar a disciplina nos períodos da manhã, da tarde e da noite.

Saída

Para cada caso de teste, imprima uma linha contendo os códigos de todas as turmas a serem criadas, em maiúsculas e separados por um espaço em branco. Imprimir os códigos das turmas em ordem crescente de seus valores.

Exemplo de entrada

```
2
ED 07 40 50 20 219
Po 23 25 0 20 119
```

Exemplo de saída

```
ED1A07 ED1B07 ED2J07 ED3P07 ED3Q07 ED3R07 ED3S07 ED3T07 ED3U07
PO2J23 PO3P23 PO3Q23 PO3R23 PO3S23 PO3T23
```

Problema I

Irrigação

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

Autor: Leandro Luque (Fatec Mogi das Cruzes)

Um importante fator relacionado à sobrevivência e produtividade de plantações é a irrigação. Entre as diferentes formas de irrigação, uma das mais utilizadas é a aspersão, por meio da qual a água é distribuída quase uniformemente em todo o terreno. Este tipo de irrigação é muito utilizado em residências, geralmente através de um aspersor giratório que lança água em todas as direções, dentro de um raio de alcance.

Valtuir trabalha com a instalação de sistemas de irrigação por aspersão e precisa definir a melhor posição para um aspersor de alcance circular que consegue irrigar uma área dentro de um raio *R*. A escolha da posição do aspersor depende das características da plantação e do custo. Como Valtuir recebe de Girdácio, seu chefe, uma comissão por cada real economizado na instalação do sistema, ele deseja encontrar a posição e o alcance do aspersor de forma a minimizar os custos de instalação. Existem aspersores com raio de alcance (*R*) múltiplo de 5 (5m, 10m, 15m, 20m, ...) e seu custo é proporcional ao raio, por meio da fórmula: R*100-100/R.

Neste problema, você deve ajudar Valtuir a definir a posição ideal de instalação de um aspersor, bem como seu raio de alcance, a partir de informações sobre a área de plantação e a posição das plantas que devem ser irrigadas. Todas as medidas são informadas em metros.

Entrada

A entrada é composta por diversos casos de teste. A primeira linha de cada caso de teste contém dois números inteiros C ($0 < C \leq 1000$) e L ($0 < L \leq 1000$), separados por um espaço, que indicam o comprimento e a largura da área de plantação, respectivamente. A próxima linha contém um número inteiro N ($0 < N \leq 1000$), representando o número de plantas desta plantação. As N linhas seguintes contém dois números inteiros X ($0 < X < C$) e Y ($0 < Y < L$), separados por um espaço, representando a posição de cada planta dentro da plantação, assumindo coordenadas cartesianas e também que o canto inferior esquerdo da plantação está posicionado em (0,0). A última linha da entrada contém os valores "0 0".

Saída

Para cada caso de teste, imprima dois números reais (com dois dígitos decimais de precisão) e um número inteiro, separados por espaço, representativos da posição x, y e do raio de alcance do aspersor, de tal forma que todas as plantas sejam por ele irrigadas e o custo seja o mínimo possível.

Exemplo de entrada

```
10 10
2
3 3
7 7
0 0
```

Exemplo de saída

```
5.00 5.00 5
```