Problem A. A Sequência de Simbanacci

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

O Rei Lennôn sempre quis atingir as expectativas de pai. Ele procurou entender como funciona o ciclo da vida desde cedo, estudando um organismo unicelular que batizou de *Starr* (em homenagem a seu tio). Após análises, ele percebeu um ritmo muito peculiar na reprodução dos bichinhos!

- um *Starr* demora 2 horas para amadurecer;
- um *Starr* maduro gera 3 novos organismos a cada hora.
- um *Starr* só vive 4 horas...

O Rei Lennôn ficou curioso sobre isso, e lhe chamou para calcular o quanto eles crescem ao longo do tempo.

Input

A entrada consiste em dois números inteiros S e H, $0 \leq S \leq 42$ e $0 \leq H \leq 10^4$, separados por espaço, representando a população inicial de *Starrs* (recém nascidos!) e a quantidade de horas do experimento, respectivamente.

Output

Apresente a quantidade de *Starrs* vivos após H horas decorridas.

Examples

standard input	standard output
1 8	36
3 3	12
40 40	29389868280

Note

No primeiro exemplo, o único *Starr* reproduz passadas as terceira e quarta horas (infelizmente falecendo em seguida). A primeira ninhada (de 3 *Starrs*) segue o mesmo ritmo e produz uma nova geração (de 9 *Starrs*) três horas depois de ter nascido, portanto serão 15 *Starrs* no experimento neste momento (o primeiro *Starr* gerou uma segunda ninhada). Após mais uma hora, as 2 ninhadas da primeira geração se reproduzem, mas o tempo se esgota para a mais velha delas, que padece. Passada a última hora, a segunda ninhada da primeira geração produz mais uma ninhada e morre, restando 36 *Starrs* vivos.

No segundo exemplo, começamos com 3 *Starrs* que se reproduzem uma vez depois da terceira hora de vida.

Problem B. Beto e DP

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Beto pretende desenvolver um software para estudo e manipulação de polinômios denominado DP (*Dynamic Polinomials*). O programa mantém três polinômios $f(x), g(x)$ e $h(x)$, onde $h(x) = f(x)g(x)$. Inicialmente, os coeficientes de todos os três polinômios são iguais a zeros e, a cada alteração promovida em $f(x)$ ou $g(x)$, $h(x)$ é atualizado automaticamente.

A interface do software deve suportar quatro comandos:

- `1` i v: adiciona v ao coeficiente a_i do polinômio $f(x) = a_0 * x^0 + a_1 * x^1 + a_2 * x^2 + \ldots$
- `2` j v: adiciona v ao coeficiente b_j do polinômio $g(x) = b_0 * x^0 + b_1 * x^1 + b_2 * x^2 + \ldots$
- `3` k: imprime o valor do coeficiente c_k do polinômio $h(x) = c_0 * x^0 + c_1 * x^1 + c_2 * x^2 + \ldots$
- `4` x_0: imprime 1 se $h(x_0)$ é impar, ou 0, se $h(x_0)$ é par.

Contudo Beto está com dificuldade em implementar este software de forma eficiente. Ajude-o nesta tarefa, escrevendo um programa que receba Q comandos dos tipos descritos e os processe na ordem dada.

Input

A primeira linha da entrada contém o inteiro Q ($1 \leq Q \leq 4 \times 10^3$).

As Q linhas seguintes contém, cada uma, um dos quatro comandos descritos no texto do problema, onde os parâmetros estão sujeitos as seguintes restrições:

- $0 \leq i, j \leq 10^9$
- $0 \leq k \leq 10^{18}$
- $-10^3 \leq v \leq 10^3$
- $-10^9 \leq x_0 \leq 10^9$

É garantido que há ao menos um comando do tipo `3` ou do tipo `4`.

Output

Para cada um dos comandos dos tipos `3` e `4`, imprima, em uma linha, a informação descrita, na ordem em que os comandos foram dados na entrada.

Example

standard input	standard output
7 1 0 6 1 1 -5 1 2 1 2 1 -1 2 3 1 3 4 4 -1	-5 0

Note

O processamento dos três primeiros comandos (tipo `1`) determinam o polinômio $f(x) = x^2 - 5x + 6$; após os dois comandos seguintes $g(x) = x^3 - x$. Neste ponto,

$$h(x) = f(x)g(x) = x^5 - 5x^4 + 5x^3 + 5x^2 - 6x$$

Assim, $c_4 = -5$ e $h(-3) = -720$.

Problem C. Cotas e Rateio Linear

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

No mundo dos investimentos existem diversas opções para os investidores, como títulos do tesouro, poupança, ações da bolsa e Fundos de Investimento Imobiliários.

Os Fundos de Investimento Imobiliários, também conhecidos como FIIs, são uma combinação de recursos destinados à aplicação em empreendimentos imobiliários. Os FIIs são constituídos sob forma de condomínio fechado, isso é, o fundo é dividido em cotas, que representam parcelas do seu patrimônio[1].

Quando surge um novo fundo é realizada uma oferta pública inicial, também conhecida como IPO (*Initial Public Offer*), onde é definida a quantidade de cotas que serão emitidas e o valor de cada cota. Qualquer interessado na compra desse ativo financeiro se manifesta por meio de uma inscrição no IPO. Porém quando a demanda supera a oferta ocorre uma anomalia de mercado, já que o preço do IPO é constante. Assim é necessário realizar uma estratégia de divisão entre os interessados, denominada rateio.

Existem vários tipos de rateio, porém o mais utilizado se chama rateio linear (*linear ratio*). Nesse tipo de divisão, os interessados são enfileirados pela ordem de inscrição, recebendo um identificador inteiro único de 1 a N. A cada etapa do rateio, o indivíduo que ocupa a primeira posição da fila adquire uma única cota e, caso tenha interesse em mais cotas, retorna para a fila, na última posição. Se ele já tiver adquirido o total de cotas que desejava, ele abandona a fila e não retorna mais.

Dado o número N de interessados e a quantidade Q de cotas emitidas, determine a quantidade de cotas adquiridas por cada participante.

Input

A primeira linha da entrada contém os valores dos inteiros N ($1 \leq N \leq 2 \times 10^5$) e Q ($1 \leq Q \leq 10^{18}$), separados por um espaço em branco.

A segunda linha contém N inteiros c_i ($1 \leq c_i \leq 10^{18}$), separados por um espaço em branco, representando o número que o i-ésimo inscrito deseja adquirir. É garantido que $c_1 + c_2 + \ldots + c_N > Q$.

Output

Imprima, em uma linha, N inteiros a_i, separados por um espaço em branco, os quais indicam o número de cotas adquiridas pelo i-ésimo inscrito.

Examples

standard input	standard output
3 5 3 1 2	2 1 2
5 10000 2000 4000 10000 10 4000	2000 2664 2663 10 2663

Note

No primeiro caso, 1 adquire a primeira cota, e retorna ao fim da fila. Em seguida, 2 adquire a segunda cota e, como obteve o que desejava, abandona a fila. Então 3 adquire a terceira e volta para a fila; 1 adquire a quarta cota e volta ao final da fila. Por fim, 3 adquire a quinta e última cota.

O segundo caso é análogo.

[1] `https://pt.wikipedia.org/wiki/Fundo_de_Investimento_Imobiliário`

Problem D. Dobra ou Paga

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Na etapa final do *game show* **Dobra ou Paga** há uma fila de N cartas, com as faces viradas para cima, onde cada face contém um premio de R_i reais. O participante deve escolher quais as cartas ele irá dobrar: se a carta i for escolhida, o valor do prêmio indicado por ela será dobrado, e as cartas vizinhas ($i-1$ e $i+1$, se existirem) serão viradas com a face para baixo, e o premio indicado por elas deverá ser pago pelo participante.

O prêmio total do participante será dado pela soma do dobros dos valores indicados pelas cartas escolhidas, subtraída pelo total das cartas a serem pagas.

Dados os valores de N e dos prêmios de cada uma das cartas, determine o valor do maior prêmio total que o participante pode obter.

Input

A primeira linha da entrada contém o valor do inteiro N ($1 \leq N \leq 2 \times 10^5$).

A segunda linha contém N inteiros R_i ($1 \leq R_i \leq 1000$), separados por um espaço em branco, os quais indicam os prêmios das i-ésimas cartas ($1 \leq i \leq N$).

Output

Imprima, em uma linha, o valor do prêmio total máximo que o participante pode obter.

Examples

standard input	standard output
3 2 1 3	9
5 1 2 100 50 1	152
5 1 1 1 1 1	4

Note

No primeiro caso, o participante deve escolher as cartas 1 e 3, e pagar a carta 2, obtendo um prêmio total de $2 \times 2 + 2 \times 3 - 1 = 9$ reais.

No segundo caso, o participante deve escolher as cartas 1, 3 e 5, e pagar as cartas 2 e 4, de modo que o prêmio total será igual a

$$2 \times 1 + 2 \times 100 + 2 \times 1 - 2 - 50 = 152$$

Observe que, ao escolher a carta 3, as cartas 2 e 4 são viradas, de modo que não é possível escolher simultaneamente as cartas 3 e 4 (se 4 fosse escolhida, 3 e 5 seriam viradas).

No terceiro caso as cartas de índice ímpar devem ser escolhidas, e as com índice par pagas.

Problem E. Eds e a Prova Perfeita

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	256 megabytes

Era uma vez cinco irmãos, que tinham talentos peculiares na elaboração de provas. Os trigêmeos mais velhos, Edsinho, Ed Filho e Edsão, se completavam pois cada um sabia como ninguém produzir questões com dificuldades diferentes. Infelizmente nunca se acertavam na hora de balancear a prova, ou era muito fácil (esse Edsinho...), ou muito difícil (quando Edsão se empolgava) ou muito monótona (se Ed Filho tomasse as rédeas).

Os gêmeos caçulas Edsoff e Edson, quando cresceram, se juntaram a eles para elaborar a prova perfeita, seguindo a simples regra do ilustre Costa Júnior: "a prova perfeita atende três propriedades: todos os times conseguem resolver pelo menos um problema; cada problema é resolvido por pelo menos um time; e nenhum time resolve todos os problemas."

Dados os resultados de uma maratona, indique se foi aplicada a prova perfeita ou não.

Input

A primeira linha da entrada contém dois inteiros P e T ($0 < P \leq 26, 1 \leq T \leq 55.000$), separados por espaço, indicando a quantidade de problemas e a quantidade de times, respectivamente.

As T linhas seguintes apresentam, cada uma, o resultado de cada time na prova, na forma de P pares de valores $\langle p, t \rangle$, onde p é uma das P primeiras letras do alfabeto, maiúscula, identificando o problema e t ($0 \leq t \leq 300$) é um valor numérico indicando o tempo, em minutos, que o time levou para acertar o problema p.

Os valores de p e t são apresentados contiguamente e os pares são separados por um espaço em branco. O tempo de resolução é sempre arredondado para o minuto acima, de modo que $t = 0$ que dizer que o time não acertou a questão.

Output

Apresente, em uma linha, a mensagem "Perfeita!" se a prova atende aos requisitos do ilustre Costa Júnior, ou "...", caso contrário.

Examples

standard input	standard output
3 4 A72 B32 C0 C0 B32 A0 B2 C1 A0 A27 B27 C0	Perfeita!
4 3 A0 B1 C0 D200 A10 B100 C300 D200 B20 C0 D75 A67	...

Note

No primeiro caso todas as regras são atendidas.

No segundo caso um dos times resolve todos os problemas da prova.

,

Problem F. F1

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

No treino classificatório para a formação do grid de largada da Fórmula 1, os pilotos percorrem a extensão do circuito o mais rápido possível. O tempo da volta é registrado por sensores localizados em três pontos específicos da pista, as três parciais que somadas determinam o tempo da volta.

Larga na *pole position* o piloto que completou a volta em menor tempo. A segunda posição fica com o segundo mais rápido, e assim por diante. Em caso de empate no tempo, cuja precisão é de milésimo de segundo, o primeiro piloto a ter registrado a marca terá a primazia.

Dados os tempos das três parciais do N pilotos que participaram da sessão classificatória, determine o grid de largada da corrida.

Fato curioso: no GP da Europa de 1997, que decidiria o campeão da temporada, Jacques Villeneuve e Michael Schumacher estavam separados na tabela por apenas um ponto. No treino classificatório, os dois marcaram exatamente o mesmo tempo: 1:21.072! Se não bastasse a coincidência, Heinz-Harald Frentzen, companheiro de Villeneuve na Williams, largou em terceiro, com o mesmo tempo de 1:21.072 que resultou na pole de Villeneuve!

Input

A primeira linha da entrada contém o número N ($1 \leq N \leq 20$) de pilotos que participaram do treino classificatório. Na sequência, são das as informações da melhor volta no treino classificatório para cada piloto, na ordem em que foram registradas.

As informações de cada piloto ocupam 2 linhas cada. A primeira delas contém o nome do piloto, formado por uma string de, no máximo, 30 caracteres alfabéticos maiúsculos ou minúsculos. A segunda linha contém o tempo de cada uma das parciais do piloto, separados por um espaço em branco. O tempo é dado no formato `ss:mmm`, com $0 \leq$ `ss` ≤ 59 e $0 \leq$ `mmm` ≤ 999.

Output

Imprima N linhas, onde a i-ésima linha contém a posição, o nome e o tempo de volta do i-ésimo piloto mais rápido. A posição de largada deve ser seguida de um ponto final e de um espaço; o nome deve ser seguido por um espaço, um traço e um espaço; e o tempo de volta deve seguir o formato `M:ss.mmm`, onde `M` é dado em minutos, `ss` em segundos (com dois dígitos) e `mmm` em milisegundos (com três dígitos).

Examples

standard input	standard output
5 Verstappen 20.252 25.654 21.602 Albon 20.380 25.816 21.739 Hamilton 20.310 25.725 21.664 Vettel 20.289 25.699 21.643 Bottas 20.362 25.792 21.720	1. Verstappen - 1:07.508 2. Vettel - 1:07.631 3. Hamilton - 1:07.699 4. Bottas - 1:07.874 5. Albon - 1:07.935
3 Villenueve 26.754 26.753 27.565 Schumacher 25.943 25.944 29.185 Frentzen 28.375 26.753 25.944	1. Villenueve - 1:21.072 2. Schumacher - 1:21.072 3. Frentzen - 1:21.072

Note

A saída do primeiro caso corresponde ao grid do GP do Brasil de 2019. Os tempos das parciais, porém, é fictício.

Problem G. Grupo do Turgão

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Turgão gosta de usar o aplicativo *Telegram*a para troca de mensagens. Certo dia criou um grupo com Mikael e naturalmente o nomeou como "eu e mikael". Algum tempo depois José foi adicional e o grupo foi renomeado para "eu mikael e jose". Sempre que mais alguem entra no grupo, esse é renomeado seguindo o mesmo padrão.

Turgão quer automatizar esse processo e ninguém melhor do que você para construir o programa principal! Considere que o grupo começa apenas com Turgão e Mikael, portando com nome "eu e mikael", você deve processar N eventos. No evento i a pessoa p_i entra no grupo e você informar qual o novo nome do grupo.

Input

A primeira linha da entrada terá um inteiro $N(1 \leq N \leq 20)$, o número de eventos a serem processados. A i-ésima das próximas N linhas terá um nome $p_i(1 \leq |p_i| \leq 20)$ em letras minúsculas. Todos os nomes serão distintos.

Output

A saída deverá ser formada por N linhas. A i-ésima deve ser o nome do grupo após processar o evento i.

Example

standard input	standard output
6 jose veras luis pedro claudio dani	eu mikael e jose eu mikael jose e veras eu mikael jose veras e luis eu mikael jose veras luis e pedro eu mikael jose veras luis pedro e claudio eu mikael jose veras luis pedro claudio e dani

Problem H. Holiday na Sucupira

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

A Floresta da Nlogonia é uma grande área de conservação ambiental, que recentemente construiu trilhas ligando diversos atrativos para receber caminhantes, ciclistas e amantes da natureza. Cada trilha pode ser percorrida em ambas direções, sendo delimitada por dois pontos de interesse do parque, que podem ser a portaria do parque, os mirantes e a cachoeira do Sucupira. Para facilitar o acesso à cachoeira do Sucupira, a administração da floresta sinalizou diversos trajetos, cada um formado por um conjunto de trilhas que levam os visitantes desde a portaria até a belíssima queda d'agua.

Dona Marileusa, que é acostumada a pedalar em trilhas na natureza, fará sua primeira visita à Floresta da Nlogonia. Ela soube que, nesse mesmo dia de visita, haverá uma excursão que vai percorrer algum dos trajetos de menor distância desde a portaria da floresta até a cachoeira do Sucupira. Como a Dona Marileusa quer curtir a floresta sozinha, parando quando necessário para descansar e para tirar algumas fotos, ela decide pedalar pelo menor trajeto que não contém nenhuma das trilhas que a excursão poderá passar para chegar até a cachoeira. Caso isso não seja possível, ela desiste de visitar a floresta.

Sabendo-se que os pontos de interesse da Floresta da Nlogonia são mapeados por números inteiros entre 1 e N, de modo que a portaria é dada pelo número inteiro 1, os mirantes entre 2 e $N-1$, e a cachoeira do Sucupira pelo inteiro N, determine a distância total percorrida pela Dona Marileusa pelas trilhas da floresta para se chegar até a cachoeira Sucupira, partindo-se da portaria do parque.

Input

A primeira linha da entrada contém dois números inteiros N e M ($2 \leq N \leq 10^4, 1 \leq M \leq min(10^5, (N * (N-1))/2)$) separados por um espaço em branco, indicando a quantidade de pontos de interesse e a quantidade de trilhas na Floresta da Nlogonia, respectivamente.

Em seguida, existem M linhas, em que cada uma descreve três números inteiros u, v e $d_{u,v}$ ($1 \leq u, v \leq 10^4, u \neq v, 1 \leq d_{u,v} \leq 10^6$) separados por espaço, indicando que o trecho de distância $d_{u,v}$ é delimitado pelos pontos de interesse u e v. É garantido que se pode acessar todos pontos de interesse da floresta.

Output

Imprima um número inteiro indicando a distância percorrida por Dona Marileusa, que compreende o menor trajeto que ela pode pedalar sem passar por alguma possível trilha utilizada pela excursão. Se não for possível que Dona Marileusa pedale nesse momento, imprima -1.

,

Examples

standard input	standard output
4 5 1 2 1 1 3 3 2 4 2 2 3 1 3 4 2	5
7 9 1 2 10 1 4 12 1 5 2 4 6 4 5 6 1 6 7 3 4 7 12 3 7 21 2 3 12	24
5 5 1 2 2 2 5 3 1 3 1 3 5 4 4 5 2	-1

Note

No primeiro exemplo de teste, temos $N = 4$, logo o ponto de interesse 1 é a portaria, os pontos de interesse 2 e 3 são mirantes e o ponto 4 é a cachoeira do Sucupira. Dona Marileusa pedala pela Floresta da Nlogonia seguindo o trajeto 1 (portaria) $\rightarrow$ 3 (mirante) $\rightarrow$ 4 (cachoeira). Portanto, a distância total percorrida por Dona Marileusa é 5.

No segundo exemplo de teste, Dona Marileusa pedala pela Floresta da Nlogonia seguindo o trajeto 1 (portaria) $\rightarrow$ 4 (mirante) $\rightarrow$ 7 (cachoeira). Portanto, a distância total percorrida por Dona Marileusa é 24.

No terceiro exemplo de teste, existe dois trajetos de menor distância desde a porteira da Floresta da Nlogonia até a cachoeira do Sucupira, $1 \rightarrow 2 \rightarrow 5$ e $1 \rightarrow 3 \rightarrow 5$. Esses trajetos são as únicas maneiras de se chegar até a cachoeira e a excursão pode percorrer qualquer um deles (e suas trilhas). Assim, não existe um trajeto alternativo que Dona Marileusa pode pedalar para chegar até a cachoeira sem ter que passar por alguma das trilhas que podem ser utilizadas pela excursão.

Problem I. Ímpar/Par Sorting

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Considere uma sequência de inteiros $a_t = \{a_1, a_2, \ldots, a_N\}$, onde $N = 2^k$ para algum k inteiro não-negativo. Ímpar/Par *Sorting* é um algoritmo que ordena a sequência $\{a_t\}$ da seguinte forma:

1. todos os elementos cujos índices são ímpares são movidos para a primeira metade, mantida a ordem relativa entre eles;
2. os elementos cujos índices são pares são movidos para segunda metade da sequência, mantida a ordem relativa entre eles;
3. os passos 1 e 2 são repetidos para as subsequências $b_i = \{b_1, b_2, \ldots, b_{N/2}\} = \{a_1, a_3, \ldots, a_{N-1}\}$ e $c_j = \{c_1, c_2, \ldots, c_{N/2}\} = \{a_2, a_4, \ldots, a_N\}$, enquanto estas subsequências tiverem dois ou mais elementos.

Abaixo segue as etapas do Ímpar/Par *Sorting* para $N = 8$:

$$\begin{array}{cccc|cccc} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 \\ a_1 & a_3 & a_5 & a_7 \parallel a_2 & a_4 & a_6 & a_8 \\ a_1 & a_5 \mid a_3 & a_7 \parallel a_2 & a_6 \mid a_6 & a_8 \\ a_1 & a_5 & a_3 & a_7 & a_2 & a_6 & a_6 & a_8 \end{array}$$

Dados os valores de N e dos elementos da sequência, ordene-a utilizando o Ímpar/Par *Sorting*.

Input

A primeira linha da entrada contém o valor de N ($1 \leq N \leq 262.144$). É garantido que N é uma potência de 2.

A segunda linha contém N inteiros a_i ($-10^9 \leq a_i \leq 10^9, 1 \leq i \leq N$), separados por um espaço em branco.

Output

Imprima, em uma linha, a sequência ordenada de acordo com o Ímpar/Par *Sorting*. Os elementos devem estar separados por um único espaço em branco, e não deve haver espaço em branco após o último elemento.

Examples

standard input	standard output
2 1 2	1 2
4 1 2 3 4	1 3 2 4
8 5 9 1 4 2 3 7 6	5 2 1 7 9 3 4 6

Note

No primeiro caso, $b_i = \{1\}$ e $c_j = \{2\}$. Como ambas subsequências tem tamanho 1, o algoritmo se encerra.

No segundo caso, após a primeira etapa $b_i = \{1, 3\}$ e $c_j = \{2, 4\}$. Na etapa seguinte estas subsequências são divididas em novas subsequências de tamanho 1 cada, sem trocar a posição relativa dos elementos, e o algoritmo é encerrado.

No terceiro caso, basta dispôr os número na ordem apresentada no exemplo do texto.

,

Problem J. José de Férias

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

José está planejando suas próximas férias. Ele tirará N dias de férias e planeja passar cada dia em alguma cidade. Se a cidade escolhida no dia i for diferente da cidade escolhida no dia $i+1$, José tem que arrumar as malas e fazer uma viagem de ônibus. Após montar seu plano inicial de viagem (uma cidade para cada dia), ele pretende fazer dois tipos de processos:

1. Dizer quantas viagens de ônibus ele fará considerando um intervalo de dias dados.
2. Trocar a cidade de destino de um dia específico.

Ajude José a processar as consultas.

Input

A primeira linha da entrada contém dois inteiros N e Q ($1 \leq N, Q \leq 10^5$), onde N indica o número de dias que José tirará de férias e Q indica o número de consultas.

A segunda linha contém N inteiros separados por espaço $D_1, ..., D_N$ ($1 \leq D_i \leq 10^9$), representando qual a cidade que José pretende viajar, inicialmente, no dia i.

As Q linhas seguintes contém, cada uma, uma consulta. Toda consulta apresenta três inteiros, separados por espaço. O primeiro valor define o tipo de consulta.

A consulta do tipo 1 é representada pelos inteiros 1 l r, com ($1 \leq l \leq r \leq N$) definindo o intervalo de dias (inclusivo).

A consulta do tipo 2 é representada pelos inteiros 2 i x, onde i ($1 \leq i \leq N$) indica qual dia que será alterado e x ($1 \leq x \leq 10^9$) indica a nova cidade que José deseja visitar nesse dia.

Output

Para cada consulta do tipo 1, imprima a resposta conforme a versão atual do planejamento.

Examples

standard input	standard output
6 7 1 2 3 4 5 6 1 1 6 2 2 1 1 1 6 2 4 3 1 3 6 2 6 5 1 2 6	5 4 2 2
5 6 1000000000 1000000000 1000000000 1000000000 1000000000 1 1 5 2 1 1 2 2 2 2 2 3 1 2 2 1 1 4	0 2

Note

No primeiro exemplo, o planejamento inicial é viajar do dia 1 ao 6 para as seguintes cidades:

dia	1	2	3	4	5	6
cidade	1	2	3	4	5	6

Para a primeira consulta (1 1 6), deve-se responder quantas viagens serão feitas dos dias 1 ao 6. A resposta é 5, pois são feitas as seguintes viagens: 1 -> 2, 2 -> 3, 3 -> 4, 4 -> 5 e 5 -> 6.

A segunda consulta (2 2 1) altera o destino do **dia 2** para a cidade 1. Logo, o novo planejamento fica:

dia	1	2	3	4	5	6
cidade	1	1	3	4	5	6

A resposta da terceira consulta (1 1 6) agora é 4. As viagens feitas são: 1 -> 3, 3 -> 4, 4 -> 5 e 5 -> 6.

A quarta consulta (2 4 3) altera o planejamento para:

dia	1	2	3	4	5	6
cidade	1	1	3	3	5	6

A resposta da quinta consulta (1 3 6) é 2, pois há apenas as seguintes viagens entre os dias 3 e 6: 3 -> 5 e 5 -> 6.

Após a penúltima consulta (2 6 5) o planejamento fica:

dia	1	2	3	4	5	6
cidade	1	1	3	3	5	5

E a resposta para a consulta final (1 2 6) fica sendo 2.

Problem K. Ki-Sorte!

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	256 megabytes

No pequeno vilarejo de Grotão das Neves corre, semanalmente, a loteria Ki-Sorte! Cada um dos moradores pode adquirir um único bilhete, e cada bilhete contém um número inteiro x entre 1 e 99 - são tão poucos moradores que há bilhetes para todos e ainda sobra...

No sábado são sorteados três números A, B e C e a cada bilhete x gera um número da sorte

$$S(x) = |x - A| + |x - B|^2 + |x - C|^3$$

Será o vencedor o portador do bilhete que gera o menor número da sorte. Como o município tem poucos recursos, o resultado demora a sair, uma vez que os números da sorte são computadores manualmente. Auxilie o povo de Grotão das Neves escrevendo um programa que, dados os valores de A, B e C, determine o número do bilhete premiado.

Input

A entrada contém uma única linha, com os inteiros A, B e C ($1 \leq A, B, C \leq 99$), separados por um espaço em branco.

Output

Imprima, em uma linha, o número do bilhete vencedor. Se dois ou mais bilhetes geram o número da sorte mínimo, imprima qualquer um deles.

Examples

standard input	standard output
1 2 3	2
50 50 50	50
89 43 77	73

Note

No primeiro caso, o bilhete 2 gera o número da sorte

$$S(2) = |2 - 1| + |2 - 2|^2 + |2 - 3|^3 = 2,$$

e nenhum outro bilhete gera número da sorte menor.

No segundo caso, o bilhete 50 gera o número da sorte $S(50) = 0$. Os demais bilhetes geram números da sorte positivos.

O terceiro caso é análogo.

,

Problem L. Luís, o lazer e a louça

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Luís joga um jogo associado a um evento por tempo limitado. Todo jogador acumulou uma quantidade de *Gold* que deve ser utilizada neste evento. O evento disponibilizará N missões especiais muito divertidas, sendo que a i-ésima missão pode ser feita pelo jogador que pagar c_i de *Gold*. Além disso, toda missão só pode ser feita no máximo uma vez.

A comunidade do jogo é muito ativa e os usuários começaram a, colaborativamente, definir a diversão d_i de cada fase bem como o tempo t_i para completá-la. Luís acumulou R *Gold* e quer desembolsar pelo menos L, dado o trabalho para acumular toda essa quantia. No entanto, Luís quer escolher as missões de forma a maximizar sua diversão por hora.

Infelizmente, agora Luís tem que lavar a louça. Ajude-o computando quais missões ele deve completar, de modo que ele possa fazê-las assim que acabar de lavar a louça! Formalmente, sua tarefa consiste em selecionar um subconjunto de índices $S \subseteq \{1, 2, ..., N\}$ com $L \leq \sum_{i \in S} c_i \leq R$ que maximize

$$\frac{\sum_{i \in S} d_i}{\sum_{i \in S} t_i}$$

.

Input

A primeira linha da entrada contém três inteiros N, L e R $(1 \leq N, L, R \leq 1000)$ indicando, respectivamente, a quantidade de missões, a quantidade mínima que Luís quer gastar e seu *Gold* total.

A i-ésima das próximas N linhas contém três inteiros d_i, t_i e $c_i (1 \leq d_i, t_i, c_i \leq 1000)$, respectivamente, a diversão, o tempo para completar e custo da i-ésima missão.

Output

A primeira linha da saída terá um número em ponto flutuante r, indicando a diversão por hora máxima.

A segunda linha da saída terá um número inteiro $k (0 \leq k \leq N)$ a quantidade de missões que Luís fará.

A terceira linha da saída terá k números inteiros, os índices das missões que Luís fará.

Caso não seja possível escolher missões nas restrições de Luís, imprima diversão por hora 0 e conjunto vazio. Observe o exemplo.

Seja *ans* a resposta do juiz. Sua resposta será considerada correta caso $\frac{|ans-r|}{max(1,ans)} \leq 10^{-6}$ e as missões escolhidas sejam consistentes com sua resposta.

Examples

standard input	standard output
5 8 12 7 1 5 2 3 3 5 2 5 4 1 1 3 4 13	4.00000000003331646070 2 1 3
5 8 12 7 1 1 2 3 2 5 2 1 4 1 1 3 4 13	0.00000000000000000000 0
7 88 171 57 75 33 79 65 33 91 36 35 98 41 32 94 92 35 65 55 61 33 71 64	1.92424242424316727096 3 3 4 6
7 35 49 423 499 21 633 33 11 111 493 9 828 866 25 502 710 17 619 361 2 337 42 3	2.42682926829235157129 3 1 2 7
4 17 18 31 49 8 37 30 4 21 35 5 22 13 4	0.78070175438682909430 3 1 2 3

Note

No primeiro exemplo, uma outra resposta possível é escolher as missões 1, 3 e 4.

No segundo exemplo, note que não é possível escolher missões de forma a gastar entre 8 e 12 de *Gold*.

Problem M. Music Player

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Nesses tempos de pandemia, Lelé, um exímio programador, estava à toa em casa e que decidiu começar um projeto ousado. Ele pensou em fazer uma API inteligente para criar *playlists* de músicas de banco de dados abertos. A ideia é a seguinte: Lelé carrega um banco de dados de músicas na memória, e toda playlist que ele criar faz referência a uma ou mais músicas desse banco.

Dentre os vários módulos que a implementação dessa ferramenta envolve, um é o módulo de inserção de música na *playlist*. Lelé pediu sua ajuda nessa tarefa específica: você deve ler um banco de dados de músicas e, dado o código da música, retornar o nome da mesma.

Input

A primeira linha contém um inteiro N ($1 \leq N \leq 10^5$) representando quantas músicas estão no banco de dados.

As N linhas seguintes contém um inteiro C ($1 \leq C \leq 10^5$), que representa o código da música, um espaço, e uma cadeia de caracteres S ($1 \leq |S| \leq 50$), composta por caracteres alfanuméricos e espaços, que representa o nome da música.

As linhas seguintes contém, cada uma, um inteiro C, que representa um código de música a ser consultada. A entrada termina com `EOF` e há, no máximo, 10^5 consultas.

Output

Para cada código de música de consulta da entrada, você deve imprimir, em uma linha, o nome da música correspondente. Se a música não for encontrada no banco, você deve imprimir "Music not found".

Example

standard input	standard output
3 42 Imagine 5 Wish you were here 104 Crazy 42 104 100 2	Imagine Crazy Music not found Music not found

Note

No exemplo, as duas últimas músicas não constam no banco de dados. As duas primeiras estão presentes e a saída corresponde aos códigos informados.