Maratona de Programação SAET 2025

Caderno de Problemas

2025

Olá!

Prova preparada com muito empenho por **Henrique Farias** e **Ricardo Oliveira** para a Semana Acadêmica de Engenharia e Tecnologia (SAET) 2025 da UTFPR-Toledo. Esperamos que gostem!

Good luck & Have Fun

Informações Gerais

Esta prova contém 19 páginas (excluindo a capa), numeradas de 1 a 19. Verifique se a prova está completa.

Entrada

- A entrada deve ser lida da entrada padrão.
- Todas as linhas da entrada, incluindo a última, terminam com o caractere de fim-de-linha (`\n`).
- Quando a entrada contém vários valores separados por espaço, existe exatamente um espaço entre dois valores consecutivos na mesma linha.

Saida

- A saida deve ser escrita na saida padrão.
- A saida deve ser impressa no formato especificado pelo problema. A saida não deve conter nenhum dado adicional.
- Todas as linhas da saida, incluindo a última, terminam com o caractere de fim-de-linha (`\n`).

Problema A. Abrindo e Fechando Parênteses

Tempo limite: 1000 ms
Memória limite: 256 MiB

Uma *string* de abre e fecha parênteses está *bem balanceada* se: ou é (seguido de uma *string* bem balanceada seguida de); ou é a concatenação de duas *strings* bem balanceadas; ou é vazia (formalmente, é uma *string* gerada pela gramática $S \to (S)|SS|\varepsilon$).

É dada uma *string* inicial s com N parênteses. Processe Q operações, onde cada operação pode ser:

- 1 l r: inverta todos os parênteses no intervalo $[l..r]$; isto é, para todo $l \leq i \leq r$, troque $s[i]$ pelo inverso de $s[i]$. O *inverso* de) é (, enquanto o inverso de (é);
- 2 l r: determine se a *substring* $s[l..r]$ está bem balanceada.

Entrada

A primeira linha contém dois inteiros N e Q ($1 \leq N, Q \leq 2 \times 10^5$), o tamanho da *string* e o número de operações. A segunda linha contém a *string* inicial com N parênteses. As próximas Q linhas descrevem uma operação cada, na forma t l r onde $t = 1$ ou $t = 2$, e $1 \leq l \leq r \leq N$.

Saída

Para cada operação com $t = 2$, imprima uma linha com `sim` se a *substring* $s[l..r]$ está bem balanceada, ou `nao` caso contrário.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
10 8 ()())()()) 2 6 9 2 1 10 1 4 6 2 1 10 1 5 8 2 3 10 2 1 4 2 2 9	sim nao sim sim nao nao

Exemplo de entrada 2	**Exemplo de saída 2**
12 2))))))((((((1 1 12 2 1 12	sim

Observações

Considere o primeiro exemplo dado. Na primeira operação, a resposta é `sim` porque $s[6..9] =$`()()` está bem balanceada. Na segunda operação, a resposta é `nao` porque $s[1..10] =$ `()())()())` não está bem balanceada.

Após a terceira operação (inverte $s[4..6]$), a *string* se torna `()((())())`. A resposta da quarta operação é `sim` porque agora $s[1..10]=$ `()((())())` está bem balanceada.

Problema B. BugNote

Tempo limite: 1000 ms
Memória limite: 256 MiB

Ricardo Yagami é um genial professor universitário de algoritmos e estruturas de dados, porém ele está entediado do jeito que anda o mundo. Os dias se passam e as mesmas notícias: alunos usando IA para fazer trabalhos de algoritmos, agradecendo ao ChatGPT ao invés de Alan Turing — algo realmente deplorável.

Porém, a vida de Ricardo mudou em certo dia, quando ele estava em sua sala e, de repente, viu um caderno preto caindo do céu pela janela.

Como já estava entediado, foi até o caderno para ver do que se tratava e começou a ler:

BugNote (Caderno do Bug)

Modo de usar:

- A pessoa cujo primeiro nome for escrito neste caderno terá seu código bugado.
- Após escrever o primeiro nome da pessoa, deve-se especificar a linha exata do código dela onde o bug ocorrerá.
- O caderno não surtirá efeito se o nome da pessoa for escrito errado. O caderno considera um nome como certo se for escrito **exatamente** como dado na entrada.
- O caderno não surtirá efeito se a linha do código escrita nele exceder o total de linhas no código real.
- Uma pessoa cuja a regra anterior vier a ocorrer poderá ter seu nome escrito novamente e ainda ter efeito, porém se o número de tentativas erradas for maior ou igual a 3, a pessoa passa a ser imune para sempre aos efeitos do BugNote.

Após testar o caderno em seu próprio código, Ricardo confirmou sua veracidade e decidiu bolar um plano: punir todos os alunos que usam IA para fazer código ao invés de aprender — e se tornar o Sênior do novo mundo!

Dado o número de alunos e o número de nomes que Ricardo escreveu no BugNote, determine quais alunos foram punidos.

Entrada

A primeira linha contém dois inteiros N e Q ($1 \leq N, Q \leq 2000$), representando respectivamente o número de alunos na turma e o número de nomes que Ricardo escreveu no BugNote.

As próximas N linhas contêm uma string s_i (o nome do aluno, que é único, não contém espaços e contém no máximo 10 caracteres) e um inteiro l_i ($1 \leq l_i \leq 10^5$), representando o número de linhas no código daquele aluno.

Em seguida, as próximas Q linhas contêm uma string t_i e um inteiro x_i, representando o nome e a linha do código que Ricardo escreveu no BugNote.

Saída

A saída deve conter um inteiro K representando o número de alunos punidos seguido de K linhas com os nomes desses alunos, na mesma ordem dada na entrada.

Caso nenhum aluno tenha sido punido, imprima `-1`.

Exemplos

Exemplo de entrada 1

```
3 7
Betinha 50
Robinson 130
Evandro 45
Betinha 49
Evandro 130
Robinsno 21
Evandro 46
Evandro 51
Ian 69
Evandro 45
```

Exemplo de saída 1

```
1
Betinha
```

Exemplo de entrada 2

```
5 12
Tanaka 125
Eduardo 143
Luci 420
Yuri 420
Euler 2718
Yuri 512
Tanoka 30
Lucy 123
Yuri 499
eduardo 1
Euler 3141
Luci 123
yuri 421
EDUARDO 1
Yuri 234
Eduardo 1
Tanaka 30
```

Exemplo de saída 2

```
4
Tanaka
Eduardo
Luci
Yuri
```

Problema C. Cabeçada

Tempo limite: 1000 ms
Memória limite: 256 MiB

Certo dia Tanaka e Henrique estavam voltando da reunião da maratona de programação do PPCI. Como neste dia estava garoando, Henrique estava com seu capuz cobrindo a cabeça, o que comprometeu parcialmente seu campo de visão, levando-o a um grande desastre. Em uma certa esquina, Henrique deu uma cabeçada com tudo em uma placa de `PARE`, mas em sua defesa, essa placa era comicamente pequena para os padrões de tamanho — tanto que ela era quase da altura dele.

Desde então, Henrique ficou traumatizado e, para qualquer destino que for, deseja evitar passar por placas que sejam ridiculamente pequenas. Dado um mapa da cidade, em que Henrique precisa visitar Q lugares em ordem (isto é, após visitar um local X, ele deve partir dele para o próximo destino), determine a menor distância que ele pode percorrer para cada destino, considerando apenas caminhos *seguros*.

Um **caminho seguro** é aquele em que, caso exista uma placa em alguma rua, sua proporção com a estatura de Henrique não seja menor que $\frac{5}{4}$ (considere a estatura de Henrique igual a $1.82\,m$). Caminhos sem placas são todos considerados seguros.

Caso não exista um caminho seguro para o próximo destino, considere como ponto de partida para o destino seguinte a última posição que ele esteve.

Entrada

A primeira linha contém três inteiros N, M e Q — o número de destinos, ruas e locais que Henrique precisa visitar, respectivamente ($2 \leq N \leq 100$, $1 \leq M \leq 2 \times 100$, $1 \leq Q \leq 100$).

As próximas M linhas contêm três inteiros X, Y, D e um ponto flutuante H, indicando que existe uma rua entre X e Y, com distância D e uma placa de altura H (caso $H = 0$, considere que não há placa nessa rua) ($1 \leq X, Y \leq N$, $1 \leq D \leq 10^9$, $0 \leq H \leq 10$, H será dado com no máximo duas casas decimais).

As próximas Q linhas contêm apenas um inteiro, representando o próximo destino que Henrique precisa alcançar a partir de seu destino atual (considere que Henrique sempre começa no destino 1).

Saída

A saída deve conter Q linhas. Para cada linha, imprima um inteiro representando a distância do caminho mais curto e seguro até o destino de Henrique. Caso não exista caminho seguro para um destino, imprima `-1`.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
4 3 3 1 4 5 0 2 4 6 2.54 3 3 1 0 2 3 1	11 -1 11

Exemplo de entrada 2	**Exemplo de saída 2**
10 9 5 1 2 4 0 1 4 9 2.05 2 4 4 1.99 1 3 4 3.23 4 9 13 0 3 5 5 2.45 3 6 19 0 4 9 4 1.11 6 8 49 2.21 4 5 8 3 2	-1 9 -1 5 8

Observações

Foto da placa do ocorrido do enunciado afins de evidenciar o quão estupidamente pequena ela é (banana como referência).

Problema D. Disco de senha

Tempo limite: 2000 ms
Memória limite: 512 MiB

Douglas está muito preocupado com um grande vazamento de senhas de sua rede social favorita. Por isso, ele decidiu que irá escolher uma nova senha para sua rede. Metódico que é, Douglas decidiu que irá escolher sua nova senha usando seu *disco de senha*, que é um disco circular contendo N letras minúsculas. A senha será escolhida da seguinte forma:

1. Douglas irá escolher alguma letra no disco para ser a primeira letra da senha;
2. A senha será formada percorrendo o disco em sentido horário, a partir da primeira letra, concatenando as letras percorridas;
3. A senha deverá ter no mínimo 1 e no máximo K letras.

Quantas senhas *distintas* podem ser formadas dessa maneira?

Entrada

A primeira linha contém dois inteiros N e K ($1 \leq K \leq N \leq 10^5$), o número de letras no disco e o tamanho máximo da senha, respectivamente. A segunda linha contém N letras minúsculas indicando as letras no disco, em sentido horário, a partir de qualquer uma delas.

Saída

Imprima uma linha com o número de senhas distintas que podem ser formadas.

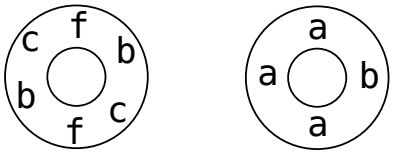
Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
`6 3` `fbcfbc`	`9`

Exemplo de entrada 2	**Exemplo de saída 2**
`4 2` `baaa`	`5`

Exemplo de entrada 3	**Exemplo de saída 3**
`6 6` `turing`	`36`

Observações



Os discos acima representam os dois primeiros exemplos de entrada. No primeiro exemplo, as senhas que podem ser formadas com até $K = 3$ letras são `b`, `c`, `f`, `bc`, `cf`, `fb`, `bcf`, `cfb` e `fbc`, totalizando 9 senhas distintas. No segundo exemplo, as senhas que podem ser formadas com até $K = 2$ letras são `a`, `b`, `aa`, `ab` e `ba`, totalizando 5 senhas distintas.

Problema E. Espaço na van

Tempo limite: 1000 ms
Memória limite: 256 MiB

Está chegando a hora! A etapa regional da Maração de Programatona ocorrerá nas próximas semanas, e o seu *coach* já está vendo como irá levar todos os N participantes da universidade para competir na cidade vizinha.

A universidade disponibiliza uma van para essas viagens. A van conta com M poltronas (além da do motorista), de diferentes larguras. Algumas poltronas são tão pequenas e apertadas que infelizmente não é possível usá-las na viagem; as poltronas que não pode ser usadas são aquelas com L cm ou menos de largura.

Sua tarefa é ajudar o seu *coach* a determinar se é possível levar todos os N participantes na van (em viagem única), ou se será necessário usar outros veículos além da van.

Entrada

A primeira linha contém dois inteiros N e M ($1 \leq N, M \leq 10^5$), o número de competidores e de poltronas na van. A próxima linha contém M inteiros l_i ($1 \leq l_i \leq 10^9$) indicando a largura de cada poltrona, em centímetros. A última linha contém um inteiro L ($1 \leq L \leq 10^9$) indicando que poltronas com L cm ou menos de largura não podem ser usadas.

Saída

Imprima uma linha com `SIM` se é possível levar todos os competidores na van, ou `NAO` caso contrário.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
4 6 300 200 350 100 400 800 200	SIM

Exemplo de entrada 2	**Exemplo de saída 2**
5 6 300 200 350 100 400 800 200	NAO

Problema F. Formação da dupla perfeita

Tempo limite: 1000 ms
Memória limite: 256 MiB

A *Respiração do Trovão* é uma técnica de luta milenar usada em combate por grandes guerreiros. Existem F *formas* diferentes de usar a técnica. Cada forma exige muito para ser dominada; por isso, alguns guerreiros podem dominar apenas algumas das formas, enquanto não dominem outras.

O Vovô é um grande mestre da Respiração do Trovão, e tem hoje N discípulos sob seu treinamento. O Vovô sabe quais formas cada discípulo seu domina, e agora quer escolher dois de seus discípulos para criar a dupla perfeita. Uma dupla é perfeita se cada forma da Respiração do Trovão é dominada por pelo menos um discípulo da dupla.

De quantas maneiras o vovô pode escolher dois discípulos para criar a dupla perfeita?

Entrada

A primeira linha contém dois inteiros N e F ($2 \leq N \leq 10^5, 1 \leq F \leq 10$), o número de discípulos e de formas da técnica, respectivamente. Os discípulos são numerados de 1 a N, e as formas são numeradas de 1 a F.

As próximas N linhas contém F caracteres cada. O j-ésimo caracter da $i-$ésima linha é `S` se o discípulo i domina a forma j, ou `N` se não domina.

Saída

Imprima uma linha com o número de maneiras de formar a dupla perfeita.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
2 6 SNNNNN NSSSSS	1

Exemplo de entrada 2	**Exemplo de saída 2**
4 4 SSNS SSSS NNNN NSSN	4

Exemplo de entrada 3	**Exemplo de saída 3**
3 3 SNN NSN NNS	0

Observações

No primeiro exemplo, a única dupla que pode ser formada é a dos discípulos $\{1, 2\}$.

No segundo exemplo, as possíveis duplas são $\{1, 2\}$, $\{1, 4\}$, $\{2, 3\}$ e $\{2, 4\}$.

Problema G. Genectorio

Tempo limite: 1000 ms
Memória limite: 256 MiB

Eduardo é um grande fã de jogos eletrônicos de automação. Recentemente, ele descobriu um novo jogo chamado *Genectorio*. Nesse jogo, o jogador recebe um lado de um filamento de DNA com N nucleotídeos, que podem ser representados pelas letras A, C, G e T.

O objetivo do jogo é construir a sequência complementar do DNA, isto é, para cada nucleotídeo:

- A deve ser pareado com T;
- T deve ser pareado com A;
- C deve ser pareado com G;
- G deve ser pareado com C.

Entretanto, para deixar o desafio mais interessante, o jogo possui um modo secreto onde a sequência complementar é criptografada utilizando uma variação da *Cifra de César*. A Cifra de César consiste em substituir cada letra da string pela letra que ocorre k posições depois dela no alfabeto (circular), onde k é o *deslocamento* da letra.

Nesse modo, o deslocamento de cada nucleotídeo é determinado por operações de `XOR` bit a bit ($\oplus$) da seguinte forma:

- Seja X um número inteiro dado na entrada.
- Para o primeiro nucleotídeo N_1, o deslocamento é calculado como $(X \oplus N_1)$.
- Para o segundo nucleotídeo N_2, o deslocamento é calculado como $(X \oplus N_1 \oplus N_2)$.
- O processo continua seguindo essa lógica cumulativa até o último nucleotídeo.

Após aplicar todas as substituições, o jogador deve imprimir a sequência complementar criptografada.

Considere que o alfabeto tem apenas as letras A,C,G e T, nesta ordem. Ainda, as letras A, C, G e T devem ser convertidas para 0, 1, 2 e 3 respectivamente para fins do cálculo do operador `XOR`.

Entrada

A primeira linha contém dois inteiros N e X ($1 \leq N \leq 10^5$, $0 \leq X \leq 10^5$). A segunda linha contém uma **string** S de tamanho N, representando a sequência de nucleotídeos.

Saída

Imprima uma linha com a **string criptografada** resultante após aplicar todas as operações descritas.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
6 7 GCATGC	GGTTGG

Exemplo de entrada 2	**Exemplo de saída 2**
20 5 AACGTATTGCATGCAGTCGG	AAGTCAGCAACCAACCTACT

Observações

No primeiro exemplo de entrada, a string complementar é `CGTACG`. O deslocamento da primeira letra é $7 \oplus 2 = 5$, e logo a primeira letra criptografada é C $\rightarrow$ G $\rightarrow$ T $\rightarrow$ A $\rightarrow$ C $\rightarrow$ G. O deslocamento da segunda letra é $7 \oplus 2 \oplus 1 = 4$, e logo a segunda letra criptografada é G $\rightarrow$ T $\rightarrow$ A $\rightarrow$ C $\rightarrow$ G.

Problema H. Hora da Pizza

Tempo limite: 1000 ms
Memória limite: 256 MiB

Nada como uma agradável noite com seus amigos na sua casa comendo uma pizza enquanto assistem seu *anime* favorito!

Opa, a pizza chegou! Como o grupo tem 4 pessoas ao todo, você decidiu cortar a pizza em 4 pedaços. Você fez um corte horizontal e outro corte vertical, de tal forma que os cortes intersectam em um ponto P na pizza.

Oh não! Você acabou de perceber que os 4 pedaços podem ter ficado de tamanho diferentes! Dados o raio da pizza e a coordenada do ponto P, determine a área dos 4 pedaços obtidos.

Entrada

A entrada contém uma linha com três inteiros R, X e Y ($1 \leq R \leq 30, \sqrt{X^2 + Y^2} < R$), o raio da pizza e as coordenadas do ponto P, respectivamente. Considere que a pizza está centrada na origem (isto é, seu centro é o ponto $(0, 0)$).

Saída

Imprima uma linha com quatro valores A_1, A_2, A_3 e A_4 indicando a área de cada pedaço da pizza, arredondadas com três casas decimais. Imprima os valores em ordem não-decrescente (isto é, de forma que $A_1 \leq A_2 \leq A_3 \leq A_4$).

Exemplos

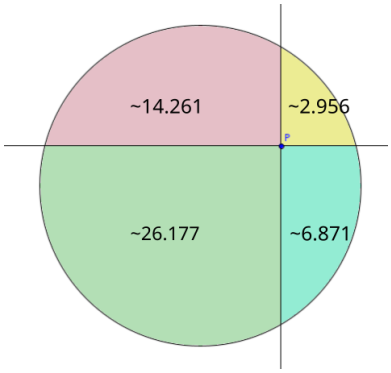
Exemplo de entrada 1	**Exemplo de saída 1**
`4 2 1`	`2.956 6.871 14.261 26.177`

Exemplo de entrada 2	**Exemplo de saída 2**
`25 -4 -10`	`188.139 307.282 594.466 873.609`

Exemplo de entrada 3	**Exemplo de saída 3**
`1 0 0`	`0.785 0.785 0.785 0.785`

Observações

A figura abaixo representa o primeiro exemplo de entrada:



Problema I. Its Over

Tempo limite: 3000 ms
Memória limite: 512 MiB

Betinha da Silva é um estudante que acha arrays de números inteiros algo muito *Pogger* e, por isso, ele coleciona arrays de números inteiros. Toda semana Betinha vai na loja de arrays e compra N arrays para adicionar na sua coleção.

Porém, Betinha é um colecionador muito *RedPill*, portanto ele só coloca em sua coleção um array se ele for um array *Banger*.

Um array é considerado um array *Banger* se a soma de todos os números nas posições primas do array também for um número primo.

Betinha acha totalmente *Brutal* ter que verificar se os arrays que ele compra são *Bangers* ou não, então ele compra todos sem mais nem menos. Mas às vezes ele pode ser *Moggado* por essa decisão: isso acontece quando ele acaba comprando um conjunto de arrays que não contém nenhum *Banger*, de forma que sobra nada para Betinha colocar em sua coleção naquela semana.

Sua tarefa é escrever um programa que determine o número de arrays *Bangers* que Betinha conseguiu naquela semana.

Entrada

A primeira linha contém um inteiro N ($1 \leq N \leq 10^3$), o número de arrays que Betinha comprou na loja naquela semana.

Em seguida, para cada um dos N arrays:

A primeira linha contém um inteiro M ($1 \leq M \leq 10^3$) , o tamanho do array.

A linha seguinte contém M inteiros A ($1 \leq A \leq 10^5$), os elementos do array. Considere que o array é 1-indexado.

Saída

A saída deve ser na primeira linha, um inteiro B, a quantidade de arrays *Bangers* que Betinha conseguiu naquela semana.

Na linha seguinte, B inteiros representando as somas correspondentes de cada array *Banger*, na ordem dada na entrada.

Caso Betinha tennha sido *Moggado*, ou seja, caso $B = 0$, a saída deve ser a seguinte frase em letras maiúsculas: ITS OVER SOBROU NADA PRO BETINHA

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
3 5 1 9 4 20 4 6 2 4 6 8 10 12 3 1 2 3	2 17 5

Exemplo de entrada 2	**Exemplo de saída 2**
2 5 2 2 2 2 2 4 5 7 3 7	ITS OVER SOBROU NADA PRO BETINHA

Observações

Na semana do primeiro exemplo Betinha tem no primeiro array, nas posições primas 2, 3 e 5, os números 9, 4 e 4 respectivamente, o que dão soma 17, que é um número primo, logo sendo um array *Banger*.

Problema J. Jogo

Tempo limite: 1000 ms
Memória limite: 256 MiB

"O Jogo"é um jogo bem peculiar de se jogar, onde basicamente você só está ganhando quando esquece completamente que ele existe, e está perdendo enquanto está consciente do jogo (Logo, você leitor está perdendo o jogo neste exato momento).

Erick é um cara muito organizado, e decidiu anotar em um caderno todas as datas em que se lembrou da existência do jogo e, consequentemente, perdeu. Agora, ele quer saber, dado esse histórico, qual foi o menor e o maior intervalo de tempo (em dias) que conseguiu ficar sem perder o jogo.

Entrada

A primeira linha da entrada contém um inteiro N ($3 \leq N \leq 10^5$), representando o número de datas registradas.

Cada uma das próximas n linhas contém três inteiros D, M, A ($1 \leq D \leq 30$, $1 \leq M \leq 12$, $1 \leq A \leq 10^9$), representando o dia, o mês e o ano de uma data em que Erick perdeu o jogo. É garantido que as datas estão em ordem cronológicas.

Considere que todos os meses tem 30 dias e todos os anos 360 dias;

Saída

A saída deve conter dois inteiros: o menor intervalo e o maior intervalo, em dias, entre duas datas consecutivas em que Erick perdeu o jogo.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
3 13 9 2025 15 9 2025 17 10 2025	2 32

Exemplo de entrada 2	**Exemplo de saída 2**
6 12 3 2019 5 11 2019 1 1 2020 4 6 2022 30 4 2025 29 9 2025	56 1046

Exemplo de entrada 3	**Exemplo de saída 3**
3 14 10 2017 23 1 2018 29 5 2024	99 2286

Problema K. Kurt, O camaleão que curte relógios

Tempo limite: 1000 ms
Memória limite: 256 MiB

Kurt, o camaleão (Mascote do PPCI) é um colecionador de relógios antigos. Ele possui N relógios, cada um marcando uma hora diferente em mostradores de 12 horas, com ponteiros de horas, minutos e segundos.

Kurt descobriu um botão curioso em cada relógio: ao pressionar o botão em um relógio específico, todos os outros relógios (exceto o pressionado) adiantam exatamente um segundo.

Por exemplo, se há três relógios marcando:

```
[01:00:00, 03:00:00, 05:00:00]
```

e Kurt pressiona o botão do segundo relógio (03:00:00), o resultado será:

```
[01:00:01, 03:00:00, 05:00:01]
```

Kurt deseja que todos os relógios mostrem exatamente a mesma hora (mesmo valor de horas, minutos e segundos). Determine o número mínimo de vezes que ele precisa apertar algum botão para que isso aconteça.

Os relógios operam em formato de 12 horas, isto é, após `11:59:59` vem `00:00:00` novamente.

Entrada

A primeira linha contém um inteiro N ($1 \leq N \leq 10^5$), o número de relógios.

Cada uma das próximas N linhas contém três inteiros h_i, m_i e s_i ($0 \leq h_i \leq 11$, $0 \leq m_i, s_i \leq 59$), representando a hora, minuto e segundo mostrados no i-ésimo relógio.

Saída

Imprima um único inteiro, o número mínimo de vezes que Kurt precisa pressionar algum botão para que todos os relógios fiquem iguais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
2 3 14 15 1 41 42	5553

Exemplo de entrada 2	**Exemplo de saída 2**
3 0 1 4 11 45 54 11 59 59	1755

Exemplo de entrada 3	**Exemplo de saída 3**
4 0 0 0 0 0 0 0 0 0 0 0 1	1

Observações

No primeiro caso teste como só há apenas 2 relógios:

Kurt pode apertar o botão do primeiro relógio até que o segundo fique sincronizado com ele, o que ocorre após 5553 cliques:

Problema L. Laranja

Tempo limite: 1000 ms
Memória limite: 256 MiB

Hicard adora laranjas, tanto que todo dia no RU (Restaurante Universitário) ele sempre pega uma laranja de sobremesa e leva para casa para poder saciar-se mais tarde. Porém, uma laranja por dia não é o bastante para ele. Por isso, seus amigos da faculdade sempre pegam a laranja a que têm direito e doam para que Hicard possa levar mais de uma laranja para casa.

Além de ser um grande fã de laranjas, Hicard também é apaixonado por análise combinatória. Considere que Hicard tem inicialmente 0 laranjas. Então, após cada dia i, ele ganha G_i novas laranjas e então come C_i laranjas.

Ele gostaria de saber, ao final de cada dia, de quantas maneiras pode rearranjar as laranjas que ainda tem em sua fruteira, que possui exatamente 3 gavetas, cada uma podendo conter até 4 laranjas. Tanto as laranjas quanto as gavetas são distinguíveis entre si, mas, dentro de uma gaveta, a ordem em que as laranjas ficam é irrelevante.

Entrada

A primeira linha contém um inteiro N ($1 \leq N \leq 10^4$), o número de dias que Hicard pegou ou comeu laranjas.

As próximas N linhas contêm dois inteiros G_i e C_i ($0 \leq G, C \leq 100$), representando o número de laranjas que Hicard ganhou e comeu em cada dia, respectivamente.

Saída

A saída deve conter N linhas. Em cada linha, imprima o número de maneiras diferentes que Hicard pode organizar sua fruteira ao final do respectivo dia. Se houver mais laranjas do que espaço na fruteira em determinado dia, isto é, se Hicard ter mais que 12 laranjas no final de um dia, imprima -1 para aquele dia. É garantido que que o número de laranjas ao final de um dia nunca será negativo.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
9 2 0 8 4 1 4 0 3 1 0 51 45 30 9 1 22 15 20	9 690 27 1 3 1890 -1 1890 9

Observações

No exemplo de entrada, Hicard tem 2 laranjas ao final do primeiro dia (por exemplo, a laranja A e a B). As 9 maneiras de rearranjá-las nas 3 gavetas são $(\{A, B\}, \{\}, \{\})$, $(\{\}, \{A, B\}, \{\})$, $(\{\}, \{\}, \{A, B\})$, $(\{A\}, \{B\}, \{\})$, $(\{A\}, \{\}, \{B\})$, $(\{B\}, \{A\}, \{\})$, $(\{B\}, \{\}, \{A\})$, $(\{\}, \{A\}, \{B\})$ e $(\{\}, \{B\}, \{A\})$.

Problema M. Madrugada na Praia

Tempo limite: 1000 ms
Memória limite: 256 MiB

Henrique "Gubi" é um garoto que adora Maratonas de Programação! Certa noite, depois de uma longa e divertida Maratona, Gubi e seus amigos (que também são maratonistas) tiveram uma ideia: passar toda a madrugada acordados na praia, curtindo a areia, o mar e o luar. Gubi topou na hora! Afinal, ele já está acostumado a virar as noites acordado, e agora fará isso com seus amigos em um lugar *good vibes*.

Gubi e seus amigos decidiram levar para a praia todos os B balões que conquistaram na Maratona daquele dia. Em um determinado momento da noite, entretanto, uma forte ventania levou V desses balões embora.

No final da noite (já ao amanhecer), Gubi e seus amigos resolveram dividir *todos* os balões que sobraram entre as N pessoas do grupo, de forma que todas as pessoas ficassem com a mesma quantidade de balões.

Ajude Gubi a determinar se é possível dividir todos os balões que sobraram igualmente entre as pessoas do grupo e, se sim, com quantos balões cada pessoa vai ficar.

Entrada

A entrada contém três inteiros N, B e V ($1 \leq N \leq 50$, $1 \leq B \leq 2000$, $0 \leq V \leq B$) indicando o número de pessoas no grupo, o número total de balões inicialmente na praia e o número de balões levados pelo vento, respectivamente.

Saída

Imprima uma linha com o número de balões com que cada pessoa irá ficar. Se não for possível fazer a divisão, imprima `-1`.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
`5 14 4`	`2`

Exemplo de entrada 2	**Exemplo de saída 2**
`7 50 1`	`7`

Exemplo de entrada 3	**Exemplo de saída 3**
`6 12 3`	`-1`