Competidor(a): _____

Número de inscrição: _____–_____ *(opcional)*

Este Caderno de Tarefas não pode ser levado para casa após a prova. Após a prova entregue este Caderno de Tarefas para seu professor guardar. Os professores poderão devolver os Cadernos de Tarefas aos competidores após o término do período de aplicação das provas (13 de setembro de 2024).

Olimpíada Brasileira de Informática

Competição Feminina - OBI2024

Caderno de Tarefas

Modalidade Programação • Nível 1 • Fase Única

13 de setembro de 2024

A PROVA TEM DURAÇÃO DE 3 HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

FUNDAÇÃO BEHRING

Coordenação:

INSTITUTO DE COMPUTAÇÃO UNICAMP

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 14 páginas (não contando a folha de rosto), numeradas de 1 a 14. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta ao *help* do ambiente de programação se este estiver disponível.
- As tarefas têm o mesmo valor na correção.
- A correção é automatizada, portanto siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa; em particular, seu programa não deve escrever frases como "Digite o dado de entrada:" ou similares.
- Não implemente nenhum recurso gráfico nas suas soluções (janelas, menus, etc.), nem utilize qualquer rotina para limpar a tela ou posicionar o cursor.
- As tarefas **não** estão necessariamente ordenadas, neste caderno, por ordem de dificuldade; procure resolver primeiro as questões mais fáceis.
- Preste muita atenção no nome dos arquivos fonte indicados nas tarefas. Soluções na linguagem C devem ser arquivos com sufixo *.c*; soluções na linguagem C++ devem ser arquivos com sufixo *.cc* ou *.cpp*; soluções na linguagem Pascal devem ser arquivos com sufixo *.pas*; soluções na linguagem Java devem ser arquivos com sufixo *.java* e a classe principal deve ter o mesmo nome do arquivo fonte; soluções na linguagem Python 3 devem ser arquivos com sufixo *.py*; e soluções na linguagem Javascript devem ter arquivos com sufixo *.js*.
- Na linguagem Java, **não** use o comando *package*, e note que o nome de sua classe principal deve usar somente letras minúsculas (o mesmo nome do arquivo indicado nas tarefas).
- Para tarefas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada tarefa.
- Ao final da prova, para cada solução que você queira submeter para correção, copie o arquivo fonte para o seu diretório de trabalho ou pen-drive, conforme especificado pelo seu professor.
- Não utilize arquivos para entrada ou saída. Todos os dados devem ser lidos da entrada padrão (normalmente é o teclado) e escritos na saída padrão (normalmente é a tela). Utilize as funções padrão para entrada e saída de dados:
  - em Pascal: *readln, read, writeln, write*;
  - em C: *scanf, getchar, printf, putchar*;
  - em C++: as mesmas de C ou os objetos *cout* e *cin*.
  - em Java: qualquer classe ou função padrão, como por exemplo *Scanner, BufferedReader, BufferedWriter* e *System.out.println*
  - em Python: *read,readline,readlines,input,print,write*
  - em Javascript: *scanf, printf*
- Procure resolver a tarefa de maneira eficiente. Na correção, eficiência também será levada em conta. As soluções serão testadas com outras entradas além das apresentadas como exemplo nas tarefas.

Bibi e a árvore

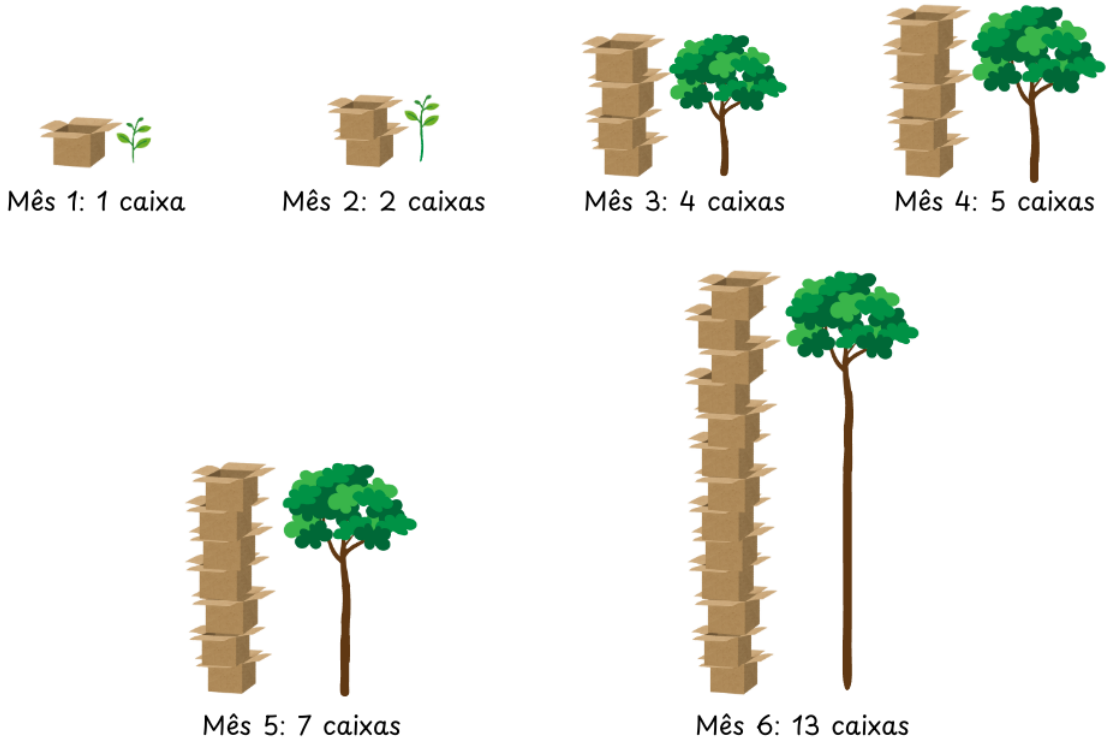
Nome do arquivo: `bibi.c`, `bibi.cpp`, `bibi.java`, `bibi.js` *ou* `bibi.py`

Bibi é uma menina apaixonada por natureza. Influenciada por sua mãe, que possui um lindo jardim botânico, ela começou desde cedo a plantar e cuidar de diversos tipos de plantas.

Bibi é extremamente curiosa e sonha em um dia ser uma grande cientista. Por isso, ela registra e acompanha, de forma independente, o crescimento de todas as plantas do jardim, anotando tudo em um livro. Ela também é bastante engenhosa com as ferramentas que possui à sua disposição: como ela não possui uma fita métrica, ela mede a altura das plantas usando caixinhas de papelão que estão prestes a ir para a reciclagem.

Num belo dia, sua mãe lhe trouxe de presente uma semente de *Abratibum*, uma árvore que supostamente vive até 100.000 anos (1.200.000 meses) e que seria a de maior altura já registrada no livro de Bibi. *Uau!*

Bibi começou imediatamente a registrar o crescimento mensal da árvore. Nos 6 primeiros meses, ela obteve os seguintes resultados:



Depois do sexto mês, Bibi observou que o crescimento da árvore parecia ter ficado fixo, e logo ela imaginou que isso poderia ser verdade para todos os meses seguintes. Deste modo, ela fez a seguinte anotação em seu caderno:

Quando a Abratibum chega aos 5 *meses de vida, ela atinge a vida adulta. Por isso, a partir de seu sexto mês de vida, a cada mês a Abratibum crescerá em altura a* ***mesma quantidade*** *de caixinhas que cresceu entre o seu quinto e o seu sexto mês de vida.*

Bibi notou que não terá caixinhas o suficiente para medir a Abratibum por muitos meses. Por isso, decidiu começar a registrar previsões de crescimento da árvore.

Sua tarefa é, dado um inteiro X, determinar a altura da árvore (em quantidade de caixinhas de papelão) no X-ésimo mês de vida dela.

Entrada

A entrada contém um único inteiro X, o mês de vida da Abratibum a ser consultado.

Saída

Seu programa deve produzir uma única linha contendo um único inteiro, a altura (em caixinhas de papelão) da Abratibum no seu X-ésimo mês de vida.

Restrições

- $1 \leq X \leq 1\,200\,000$

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (23 pontos):** $X \leq 7$.
- **Subtarefa 3 (27 pontos):** $5 \leq X \leq 12\,000$.
- **Subtarefa 4 (31 pontos):** $5 \leq X \leq 120\,000$.
- **Subtarefa 5 (19 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
1	1

Exemplo de entrada 2	**Exemplo de saída 2**
5	7

Fefe e o jogos dos monstrinhos

Nome do arquivo: `monstrinhos.c`, `monstrinhos.cpp`, `monstrinhos.java`, `monstrinhos.js` *ou* `monstrinhos.py`

Fefe acabou de descobrir um novo jogo de computador, chamado *Fadas versus Monstrinhos*.

Neste jogo existem as fadas e os monstrinhos, que são adversários. Cada fada e cada monstrinho possui uma quantidade de *poder*, que representa o quão forte ela ou ele é. Quando uma fada e um monstrinho se enfrentam, o mais forte sempre vence. Se a fada e o monstrinho tiverem a mesma quantidade de poder, é declarado um empate, isto é, ninguém perde e ninguém ganha.

Fefe possui F fadas e irá passar por um torneio onde enfrentará N monstrinhos. No jogo original, cada uma das fadas de Fefe pode enfrentar no máximo um monstrinho, pois as fadas se cansam muito rápido de tanto voar. Ou seja, se Fefe usa sua fada favorita f para derrotar um monstrinho m_0, ela não pode usar a mesma fada f novamente para derrotar um outro monstrinho m_1.

Fefe tem grande talento para modificar jogos de computador usando seu grande conhecimento em programação. Ela, então, decidiu modificar o jogo original com o objetivo de deixá-lo mais divertido. Em *Fadas versus Monstrinhos: Versão da Fefe*, cada fada de Fefe possui um *bônus*, nome dado à quantidade de monstrinhos distintos que tal fada consegue enfrentar. Por exemplo, uma fada com bônus de valor dois pode enfrentar dois monstrinhos distintos.

Fefe está determinada a derrotar o máximo de monstrinhos possível na versão que ela criou do jogo!

Dados os poderes dos monstrinhos que Fefe irá enfrentar e os poderes e bônus das fadas que Fefe possui, sua tarefa é descobrir quantos monstrinhos Fefe irá derrotar se ela jogar da melhor forma possível.

Entrada

A primeira linha da entrada contém um inteiro N, a quantidade de monstrinhos que Fefe irá enfrentar.

A segunda linha da entrada contém um inteiro F, a quantidade de fadas que Fefe possui.

As próximas N linhas representam os poderes dos monstrinhos que Fefe irá enfrentar. A i-ésima destas linhas possui o inteiro E_i, o poder do i-ésimo monstrinho.

As próximas F linhas representam os poderes das fadas de Fefe. A j-ésima destas linhas possui o inteiro P_j, o poder da j-ésima fada.

As últimas F linhas representam os bônus das fadas de Fefe. A j-ésima destas linhas possui o inteiro B_j, o bônus da j-ésima fada.

Saída

Seu programa deverá produzir uma única linha contendo um único inteiro, o número máximo de monstrinhos que Fefe consegue derrotar.

Restrições

- $1 \leq N \leq 1\,500$
- $1 \leq F \leq N$
- $1 \leq E_i \leq 1\,000\,000$ para todo $1 \leq i \leq N$

- $1 \leq P_j \leq 1\,000\,000$ para todo $1 \leq j \leq F$
- $1 \leq B_j \leq 1\,000\,000$ para todo $1 \leq j \leq F$

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (11 pontos):** $N = 1$, ou seja, Fefe irá enfrentar apenas um monstrinho.
- **Subtarefa 3 (12 pontos):** $F = 1$, ou seja, Fefe possui apenas uma fada.
- **Subtarefa 4 (14 pontos):** $B_j = N$ para todo $1 \leq j \leq F$, ou seja, todas as fadas de Fefe têm bônus igual à quantidade de monstrinhos.
- **Subtarefa 5 (19 pontos):**
  - $B_j = 1$ para todo $1 \leq j \leq F$, ou seja, cada fada só pode enfrentar um único monstrinho.
  - $1 \leq E_i \leq 3$ e $1 \leq P_j \leq 3$ para todos $1 \leq i \leq N$ e $1 \leq j \leq F$, ou seja, todas as fadas e monstrinhos possuem poder entre 1 e 3.
- **Subtarefa 6 (17 pontos):** $B_j = 1$ para todo $1 \leq j \leq F$, ou seja, cada fada só pode enfrentar um único monstrinho.
- **Subtarefa 7 (27 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
3 2 2 2 4 1 3 3 1	1

Exemplo de entrada 2	**Exemplo de saída 2**
3 3 2 2 4 1 3 6 3 1 2	3

Mistura de Poções

Nome do arquivo: `mistura.c`, `mistura.cpp`, `mistura.java`, `mistura.js` *ou* `mistura.py`

Lulu é uma estudante de magia que possui uma prateleira com N poções mágicas dispostas lado-a-lado. Cada poção tem um tipo específico, sendo a_i o tipo da i-ésima poção da esquerda para a direita.

Com a chegada de seus novos lagartos de estimação, Lulu precisa liberar espaço na prateleira. Para isso, ela pretende remover uma certa quantidade de poções do canto esquerdo e uma certa quantidade de poções do canto direito da prateleira, mantendo no final um segmento contíguo de poções na prateleira.

Além disso, Lulu quer garantir que seja possível criar pelo menos um feitiço usando as poções restantes na prateleira. Para criar um feitiço, ela precisa misturar K poções de tipos diferentes, sendo K um número pequeno.

Lulu percebeu que podem existir muitos modos de liberar espaço da forma como ela deseja. Indecisa sobre como fazer isso, ela pediu para você escrever um programa que determine a quantidade de formas diferentes de remover poções dos cantos da prateleira, atendendo às suas restrições.

Entrada

A primeira linha da entrada contém dois inteiros N e K, representando a quantidade de poções na prateleira e o número de poções de tipos diferentes necessárias para criar um feitiço, respectivamente.

A segunda linha da entrada contém N inteiros a_1, a_2, ..., a_N representando os tipos de cada poção na prateleira, da esquerda para a direita.

Saída

Seu programa deverá imprimir uma única linha contendo um único inteiro, a quantidade de formas de remover poções dos cantos da prateleira atendendo às restrições de Lulu.

Restrições

- $1 \leq N \leq 100\,000$
- $2 \leq K \leq 3$
- $1 \leq a_i \leq N$ para todo $1 \leq i \leq N$

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (13 pontos):** $N \leq 100$.
- **Subtarefa 3 (17 pontos):** $N \leq 1\,000$.
- **Subtarefa 4 (29 pontos):** $K = 2$.
- **Subtarefa 5 (41 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
4 2 2 1 2 3	6

Exemplo de entrada 2	**Exemplo de saída 2**
7 3 3 1 1 5 3 7 3	12

Exemplo de entrada 3	**Exemplo de saída 3**
2 2 1 2	1

Exemplo de entrada 4	**Exemplo de saída 4**
5 3 3 2 3 3 2	0

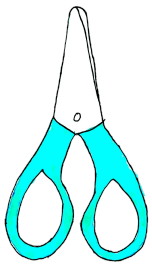
Fábrica de Tesouras

Nome do arquivo: `tesoura.c`, `tesoura.cpp`, `tesoura.java`, `tesoura.js` *ou* `tesoura.py`

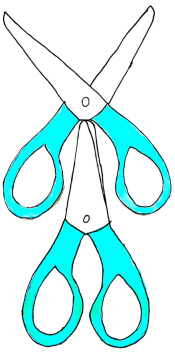
Cecília trabalha em uma fábrica de tesouras. Após o fim da super-promoção de volta às aulas, sobraram muitas tesouras, então ela decidiu usar algumas delas em um quadro para decorar a parede da fábrica. Cecília ama a natureza, e por isso decidiu que seu quadro seguirá o formato de uma árvore.

Cecília fará a montagem de acordo os passos abaixo.

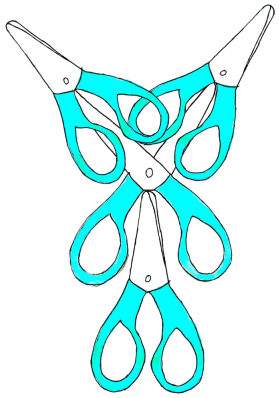
- O primeiro passo é colar uma tesoura fechada em pé na base do quadro. Esta tesoura representa o tronco da árvore:



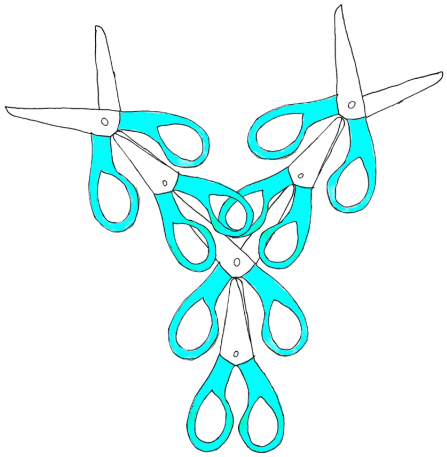
- O segundo passo é colar, na ponta das lâminas da tesoura fechada, uma tesoura aberta. Esta tesoura aberta representa os primeiros galhos da árvore:



- O terceiro passo é colar tesouras fechadas em cima de cada lâmina da tesoura posicionada no passo anterior:



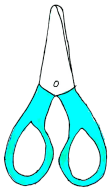
- O quarto passo é colar uma tesoura aberta na ponta das lâminas de cada tesoura colada no passo anterior:



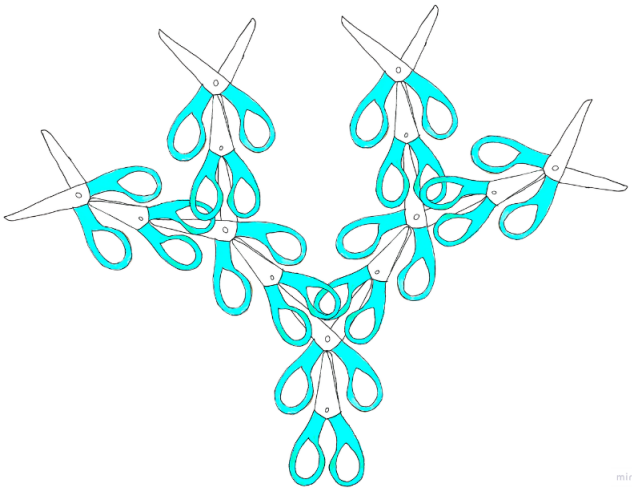
- O quinto passo segue a mesma regra do terceiro passo: colar tesouras fechadas em cima de cada lâmina das tesouras abertas coladas no quarto passo.
- O sexto passo segue a mesma regra do quarto passo: colar tesouras abertas em cima de cada lâmina das tesouras fechadas coladas no quinto passo.
- E assim em diante: Cecília vai alternando os passos entre colar tesouras fechadas e abertas até ficar satisfeita com a altura do quadro.

Cecília chama de *altura* do quadro o número de passos usados para criá-lo.

Por exemplo, o seguinte quadro tem altura 1:



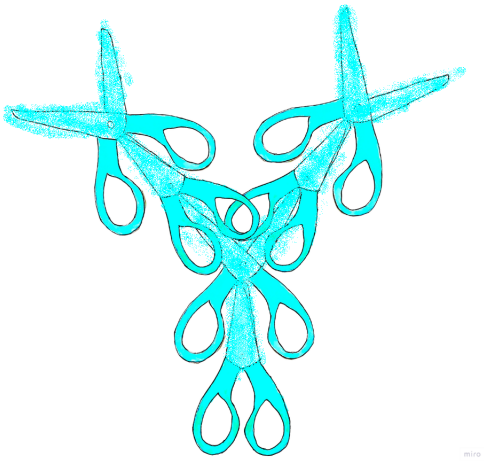
E o seguinte quadro tem altura 6:



Cecília já fez a colagem de altura K, porém ela ainda não está satisfeita com o quadro: ela está incomodada porque as lâminas das tesouras não possuem a mesma cor que os seus cabos.

Sua amiga, Laura, irá emprestar um tubo de spray de tinta para Cecília pintar a árvore com a cor dos cabos das tesouras. Porém, Laura só permitiu que Cecília use o spray uma única vez. Ou seja, Cecília não pode usar o spray para pintar uma parte da árvore, parar de pintar e recomeçar em uma outra parte da árvore.

Por exemplo, Cecília pode usar o spray para pintar a seguinte árvore, de altura 4:



Para isso, ela pode fazer o seguinte caminho com o spray:

Note que, por só poder usar o spray uma vez, Cecília teve que pintar algumas lâminas de tesouras duas vezes, gastando mais tinta.

Além disso, observe que, quando uma tesoura está fechada, uma passada de spray basta para pintar as lâminas e gastamos apenas 1 ml de tinta:

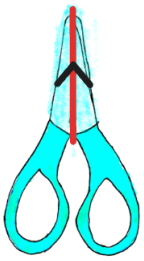


Figura 1: pintamos as lâminas em uma passada, gastando 1 ml

Por outro lado, quando uma tesoura está aberta, temos que passar o spray ao menos uma vez em cada lâmina e gastamos 1 ml de tinta em cada passada:

Figura 2: pintamos a lâmina da direita em duas passadas, gastando 3 ml no total

Cecília sempre começa pintando a árvore a partir de seu tronco (de baixo para cima) e tenta sempre

gastar o mínimo de tinta possível. Na árvore de altura 4, por exemplo, ela gastou 14 ml de tinta.

Sua tarefa é, dada a altura K da árvore que Cecília pintou, determinar a quantidade M de tinta (em ml) que ela usou. Observe que M pode ser muito grande. Se M for maior ou igual a $1\,000\,000\,007$, você deve encontrar o resto da divisão de M por $1\,000\,000\,007$.

Entrada

A única linha da entrada contém um único inteiro K, a altura da árvore montada por Cecília.

Saída

Seu programa deverá produzir uma única linha contendo um único inteiro, o resto da divisão de M por $1\,000\,000\,007$.

Restrições

- $1 \leq K \leq 50\,000$

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (7 pontos):** $K \leq 3$.
- **Subtarefa 3 (4 pontos):** $K \leq 6$ e K é par.
- **Subtarefa 4 (6 pontos):** $K \leq 6$.
- **Subtarefa 5 (19 pontos):** $K \leq 36$.
- **Subtarefa 6 (16 pontos):** $K \leq 54$ e K é par.
- **Subtarefa 7 (20 pontos):** $K \leq 54$.
- **Subtarefa 8 (15 pontos):** $K \leq 120$.
- **Subtarefa 9 (13 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
1	1

Exemplo de entrada 2	**Exemplo de saída 2**
4	14

Observação

O "resto da divisão" é o valor que sobra quando um número é dividido por outro. Por exemplo, o resto da divisão de 10 por 4 é 2. Este resto pode ser calculado utilizando o operador `%` entre o dividendo e o divisor. Por exemplo:

- O resto da divisão de 10 por 4 é calculado por `10 % 4`
- O resto da divisão de M por $1\,000\,000\,007$ é calculado por `M % 1000000007`