Competidor(a): _____

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

Olimpíada Brasileira de Informática

OBI2024

Caderno de Tarefas

Modalidade Programação • Nível Júnior • Fase 3

28 de setembro de 2024

A PROVA TEM DURAÇÃO DE 4 HORAS

Promoção:

Sociedade Brasileira de Computação

Apoio:

Coordenação:

v1.0

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.

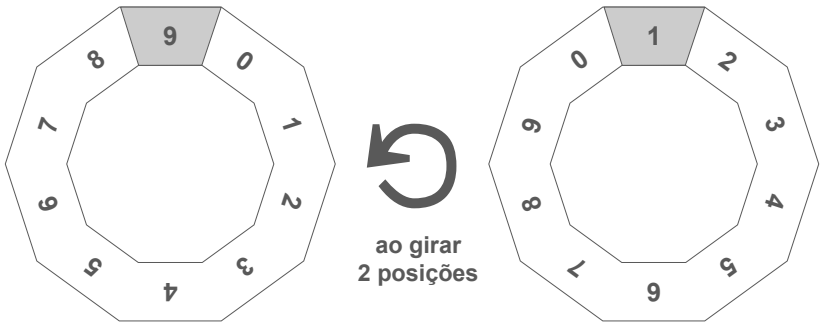
Cadeado

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

Após uma longa viagem, Fábio acabou de chegar no hotel da Semana Olímpica e quer descansar. Ele pretende tomar um banho e então vestir o pijama que está dentro de sua mala, trancada com um cadeado protegido por uma senha numérica de N dígitos.

O cadeado da mala possui N discos, um para cada dígito da senha. Cada disco possui os dígitos de `0` a `9` ordenados em sentido horário, com exatamente um desses dígitos sendo mostrado no visor do cadeado. Para trocar um dígito mostrado no visor, Fábio pode girar o disco em sentido horário ou anti-horário.

Por exemplo, considere um disco que inicialmente mostra o dígito `9`. Se girarmos esse disco em uma posição no sentido horário, ele mostrará o dígito `8`. Por outro lado, se girarmos esse disco em duas posições no sentido anti-horário, ele mostrará o dígito `1`. Conforme ilustrado nas figuras a seguir:



Cada vez que um disco é girado em uma posição em qualquer sentido, ele faz um barulho de clique.

Para abrir o cadeado, os dígitos mostrados pelos N discos devem corresponder à senha escolhida por Fábio, na mesma ordem. Por exemplo, se a senha for `081` e inicialmente os dígitos mostrados no cadeado forem `099`, Fábio pode girar o segundo disco em uma posição no sentido horário, obtendo `089`, e então girar o terceiro disco duas posições no sentido anti-horário para obter `081` e abrir o cadeado. No total, houveram 3 barulhos de cliques. Podemos verificar que é impossível mudar os

dígitos mostrados de `099` para `081` com 2 ou menos cliques.

Está muito tarde e o colega de quarto de Fábio já está dormindo. Por isso, Fábio quer evitar fazer barulho e deseja abrir o cadeado com o mínimo de cliques possível. Porém, o computador dele também está dentro da mala, e por isso ele pediu a sua ajuda.

Dada a quantidade N de dígitos na senha do cadeado, o dígito mostrado inicialmente por cada disco, e os dígitos da senha do cadeado, determine o número mínimo de cliques necessários para Fábio abrir o cadeado.

Entrada

A primeira linha da entrada contém um único inteiro N, a quantidade de discos no cadeado (que corresponde ao número de dígitos da senha).

Cada uma das próximas N linhas contém dois inteiros c_i e s_i, o dígito inicialmente mostrado no i-ésimo disco do cadeado e o i-ésimo dígito da senha, respectivamente.

Saída

Seu programa deverá produzir uma única linha contendo somente um inteiro, a quantidade mínima de cliques necessária para Fábio destravar o cadeado.

Restrições

- $1 \leq N \leq 100$
- $0 \leq c_i \leq 9$ para $1 \leq i \leq N$
- $0 \leq s_i \leq 9$ para $1 \leq i \leq N$

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 (20 pontos):** $N = 1$, ou seja, a senha é composta por apenas um dígito.
- **Subtarefa 3 (20 pontos):** $N = 5$, ou seja, a senha é composta por exatamente cinco dígitos.
- **Subtarefa 4 (30 pontos):** $c_i = 0$ para todo $1 \leq i \leq N$, ou seja, inicialmente todos os discos mostram o dígito `0`.
- **Subtarefa 5 (30 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
3 0 0 9 8 9 1	3

Explicação do exemplo 1: Este é o exemplo mostrado no enunciado: os discos inicialmente mostram os dígitos `099` e a senha é `081`; são necessários três cliques para abrir o cadeado.

Exemplo de entrada 2	**Exemplo de saída 2**
1 2 8	4

Explicação do exemplo 2: Neste caso, a senha possui apenas um dígito, `8`, e o disco do cadeado está mostrando o dígito `2`. Fábio pode mover o disco em quatro posições no sentido horário para obter o dígito `8` e abrir o cadeado.

Exemplo de entrada 3	**Exemplo de saída 3**
5 4 4 3 9 1 8 2 1 1 2	9

Exemplo de entrada 4	**Exemplo de saída 4**
10 0 9 0 8 0 7 0 6 0 5 0 4 0 3 0 2 0 1 0 0	25

Amigos

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

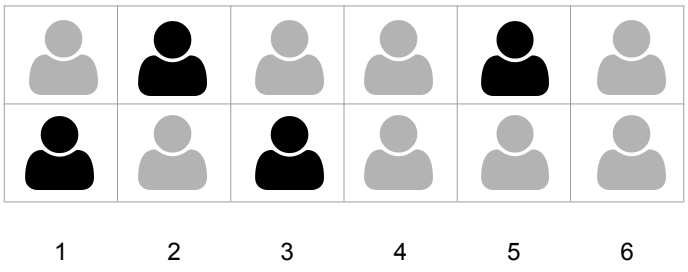
Na segunda fase da OBI, você escreveu um programa que ajudou Luiza a encontrar o ponto de ônibus mais próximo da escola dela. Agora, Luiza gasta pouco tempo no trajeto e tem mais tempo livre para fazer e cultivar amizades, com as quais ela conversa durante o intervalo da aula.

O refeitório da escola possui uma longa mesa retangular. Os lados superior e inferior da mesa possuem N cadeiras em cada (observe que não existem cadeiras nos lados esquerdo e direito da mesa). Deste modo, $2 \cdot N$ alunos podem se sentar à mesa (N de cada lado) e cada aluno senta de frente para um aluno sentado no lado oposto ao dele.

O grupo de amigos de Luiza possui um número par de integrantes, que vamos chamar de $2 \cdot K$, e eles gostariam de se sentar todos juntos à mesa. Porém, isso é muito difícil de coordenar, uma vez que o refeitório está sempre lotado. Por isso, o grupo de amigos decidiu que eles estarão felizes se cada integrante do grupo estiver sentado de frente para outro integrante do grupo. Observe que, como o número de integrantes do grupo é par, é possível que exatamente metade dos integrantes se sente em um lado da mesa e a outra metade se sente no outro lado.

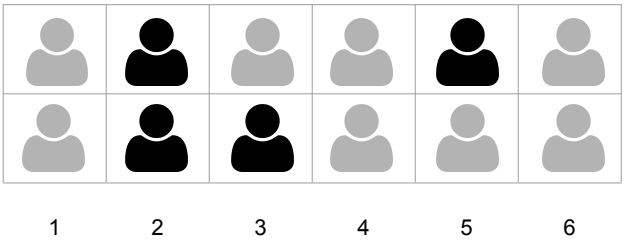
Os amigos se sentaram e já estão divididos igualmente entre os dois lados da mesa, mas nem todos os integrantes do grupo estão sentados em frente a outro integrante. Por sorte, o grupo de amigos desenvolveu um plano: um integrante pode pedir para trocar de lugar com um dos alunos sentados nas cadeiras vizinhas à sua (ou seja, os alunos sentados imediatamente à esquerda e à direita dele que estão mesmo lado da mesa, se existirem). Assim, um integrante pode se mover para a esquerda ou para a direita em seu lado da mesa usando várias trocas, sempre com um dos vizinhos atuais. O plano do grupo de amigos consiste em realizar essas trocas até que cada integrante do grupo esteja sentado em frente a outro integrante do grupo.

O diagrama abaixo ilustra um exemplo para $N = 6$ e $K = 2$, ou seja, cada lado da mesa possui 6 alunos dos quais 2 são membros do grupo. Neste exemplo, os integrantes do grupo no lado superior da mesa estão nas posições 2 e 5, já os integrantes no lado inferior da mesa estão nas posições 1 e 3.

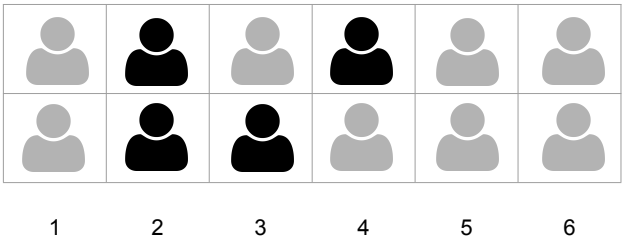


Os amigos conseguem sentar de frente para seus amigos após realizarem três trocas:

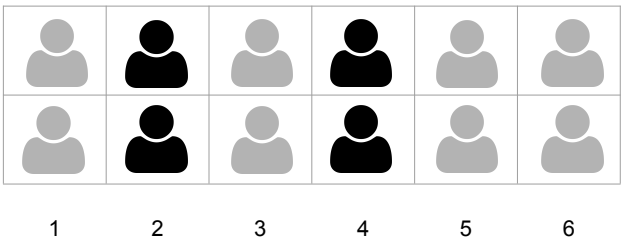
- No lado inferior da mesa, o aluno na posição 1 troca de lugar com o aluno na posição 2:



- No lado superior, o aluno na posição 5 troca de lugar com o aluno na posição 4:



- No lado inferior, o aluno na posição 3 troca de lugar com o aluno na posição 4:



Agora, em ambos os lados os amigos estão nas posições 2 e 4, e portanto eles estão de frente para seus amigos. É possível verificar que os amigos não conseguem atingir este objetivo com duas ou menos trocas.

Os amigos de Luiza querem ficar felizes sem fazer muita bagunça. Por isso, eles gostariam de fazer a menor quantidade possível de trocas para realizar o plano do grupo. No exemplo anterior, os integrantes do grupo estão reunidos em pares após 3 trocas. É possível verificar que essa é a menor quantidade possível (ou seja, não é impossîvel completar o plano do grupo com 2 ou menos trocas).

Luiza pediu a sua ajuda: dados o número N de alunos em cada lado da mesa, o número K de integrantes do grupo de amigos em cada lado da mesa, e as posições iniciais onde os integrantes do grupo estão sentados, determine o número mínimo de trocas necessárias entre alunos sentados em cadeiras vizinhas para que cada integrante do grupo de amigos sente em frente a outro integrante do grupo.

Entrada

A primeira linha da entrada contém dois inteiros N e K representando, respectivamente, o número de alunos em cada lado da mesa e o número de integrantes do grupo de amigos em cada lado da mesa.

A segunda linha da entrada descreve o lado superior da mesa e contém N inteiros, cada um sendo `0` ou `1`, separados por espaços em branco. O i-ésimo inteiro a_i indica se o aluno sentado na i-ésima cadeira (da esquerda para a direita) do lado superior da mesa é integrante do grupo de amigos ou não. Se $a_i = 1$, o aluno pertence ao grupo; caso contrário, ele não pertence. Existem exatamente K valores `1` e $N - K$ valores `0`.

A terceira e última linha da entrada descreve o lado inferior da mesa no mesmo formato da linha anterior. A linha contém N inteiros, cada um sendo `0` ou `1`, separados por espaços em branco. Se o j-ésimo inteiro b_j é `1`, o aluno sentado na j-ésima cadeira do lado inferior da mesa é integrante do grupo de amigos; caso contrário, este aluno não é integrante do grupo. Existem exatamente K valores `1` e $N - K$ valores `0`.

Saída

Seu programa deverá produzir uma única linha contendo somente um inteiro, a quantidade mínima de trocas necessárias para que todo integrante do grupo fique sentado de frente para outro integrante do grupo.

Restrições

- $2 \leq N \leq 150\,000$
- $1 \leq K < N$
- $a_i = 0$ ou 1 para $1 \leq i \leq N$
- Existem exatamente K índices i para os quais $a_i = 1$
- $b_j = 0$ ou 1 para $1 \leq j \leq N$
- Existem exatamente K índices j para os quais $b_j = 1$

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 (21 pontos):** $N \leq 1\,000$ e $K = 2$, ou seja, existem exatamente dois integrantes do grupo em cada lado da mesa.
- **Subtarefa 3 (23 pontos):** $N \leq 1\,000$.
- **Subtarefa 4 (25 pontos):** $a_i = 1$ para $i \leq K$ e $a_i = 0$ para $i > K$, ou seja, no lado superior da mesa, os integrantes do grupo estão nas primeiras K posições. *(Veja o exemplo 3.)*
- **Subtarefa 5 (31 pontos):** Sem restrições adicionais.

Exemplos

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

`6 2`
`0 1 0 0 1 0`
`1 0 1 0 0 0`
	`3`

Explicação do exemplo 1: Este é o exemplo mostrado no enunciado.

Exemplo de entrada 2	**Exemplo de saída 2**
12 4 0 0 1 0 1 1 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 1 0 1	7

Exemplo de entrada 3	**Exemplo de saída 3**
9 4 1 1 1 1 0 0 0 0 0 1 0 0 1 0 1 0 0 1	10

Exemplo de entrada 4	**Exemplo de saída 4**
4 2 1 0 0 1 1 0 0 1	0

Entrevistas de emprego

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

A Organização de Brincadeiras Infantis (OBI) está com um plano de expansão internacional e, para isso, abriu vagas para contratar novos funcionários.

O departamento de Recursos Humanos (RH) da empresa identificou N candidatos (identificados de 1 a N) para o processo seletivo, que consistirá de E entrevistas em grupo para decidir os novos contratados. Como existem vagas para diferentes departamentos da OBI, cada entrevista só é relevante para alguns dos candidatos. Deste modo, cada entrevista pode ser composta por um grupo diferente de candidatos.

O RH descobriu que alguns dos candidatos são amigos entre si. Quando dois candidatos que são amigos entre si participam da mesma entrevista, um dos amigos pode ajudar o outro. Por isso, o RH deseja evitar que dois amigos participem da mesma entrevista.

A OBI obteve uma tabela $N \times N$ que indica, para cada par de candidatos, se eles são amigos entre si ou não. Além disso, a empresa sabe que, como bons brasileiros, os candidatos seguem o velho ditado *"o amigo do meu amigo é meu amigo."* Ou seja: se os candidatos a e b são amigos e os candidatos b e c são amigos, então os candidatos a e c também são amigos.

Sua tarefa é ajudar o RH a determinar, para cada entrevista, se existe um par de candidatos convidados para a entrevista que são amigos entre si.

Entrada

A primeira linha da entrada contém um único inteiro N, o número de candidatos que estão participando do processo seletivo.

As próximas N linhas representam a tabela de amizades obtida pela OBI. Cada uma destas linhas contém N caracteres, cada um deles sendo `0` ou `1` (o dígito zero ou o dígito um). O j-ésimo caractere da i-ésima linha, m_{ij}, é `1` se os candidatos com identificadores i e j são amigos, ou `0` se eles não são amigos. Relações de amizade são sempre recíprocas, ou seja, $m_{ij} = m_{ji}$ para todos i e j. Para simplificar, a tabela não considera um candidato como amigo de si mesmo, ou seja, $m_{ii} =$ `0` para todo i. Observe que não existem espaços em branco entre os caracteres na mesma linha.

A próxima linha contém um único inteiro E, a quantidade de entrevistas a serem realizadas.

As próximas E linhas representam as entrevistas. A i-ésima destas linhas contém um inteiro K_i, indicando a quantidade de candidatos que foram convidados para a i-ésima entrevista, seguido de K_i inteiros distintos, $c_{i1}, c_{i2}, \ldots, c_{iK_i}$, indicando os identificadores dos candidatos convidados para a i-ésima entrevista.

Saída

Seu programa deverá imprimir E linhas, uma para cada entrevista, na mesma ordem da entrada. A i-ésima destas linhas deverá conter uma única letra maiúscula, que deve ser:

- `S`, se existem dois candidatos amigos entre si convidados para a i-ésima entrevista;
- `N`, se não existem dois candidatos amigos entre si convidados para a i-ésima entrevista.

Restrições

- $2 \leq N \leq 2\,500$
- $1 \leq E \leq 1\,000$
- $m_{ij} =$ `'0'` ou $m_{ij} =$ `'1'` para todos $1 \leq i \leq N$ e $1 \leq j \leq N$
- $m_{ij} = m_{ji}$ para todos $1 \leq i \leq N$ e $1 \leq j \leq N$
- $m_{ii} = 0$ para todo $1 \leq i \leq N$
- Para toda tripla (a, b, c) de identificadores distintos, se $m_{ab} = 1$ e $m_{bc} = 1$, então $m_{ac} = 1$
- $2 \leq K_i \leq N$ para todo $1 \leq i \leq E$
- $1 \leq c_{ij} \leq N$ para todos $1 \leq i \leq E$ e $1 \leq j \leq K_i$
- Em cada entrevista, os identificadores dos candidatos convidados são todos distintos

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 (29 pontos):**
  - $N \leq 500$
  - $K_i = 2$ para todo $1 \leq i \leq E$, ou seja, cada entrevista é composta por exatamente dois candidatos
- **Subtarefa 3 (31 pontos):**
  - $N \leq 500$
  - $K_i \leq 50$ para todo $1 \leq i \leq E$, ou seja, cada entrevista é composta por no máximo cinquenta candidatos
- **Subtarefa 4 (13 pontos):** Cada candidato possui no máximo um amigo.
- **Subtarefa 5 (27 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
5 01001 10001 00000 00000 11000 3 3 2 3 4 4 5 3 1 4 3 1 4 2	N S S

Explicação do exemplo 1: Neste exemplo, o processo seletivo possui $N = 5$ candidatos. A tabela indica que existe amizade entre os seguintes pares de candidatos: 1 e 2, 1 e 5, 2 e 5. Serão realizadas $E = 3$ entrevistas:

- A primeira entrevista com os candidatos 2, 3 e 4.
- A segunda entrevista com os candidatos 5, 3, 1 e 4.
- A terceira entrevista com os candidatos 1, 4 e 2.

Deste modo, podemos concluir o seguinte para cada entrevista:

- Na primeira entrevista, não existem dois candidatos amigos entre si.
- Na segunda entrevista, existem dois candidatos amigos entre si (os candidatos 1 e 5).
- Na terceira entrevista, existem dois candidatos amigos entre si (os candidatos 1 e 2).

Exemplo de entrada 2	**Exemplo de saída 2**
5 01100 10100 11000 00001 00010 5 2 2 1 2 3 4 2 2 3 2 4 2 2 5 4	S N S N S

Explicação do exemplo 2: Existem candidatos amigos entre si nas entrevistas de números 1 (pois 1 e 2 são amigos), 3 (pois 2 e 3 são amigos) e 5 (pois 5 e 4 são amigos). Nas entrevistas de números 2 e 4, os candidatos convidados não são amigos entre si.

Exemplo de entrada 3	**Exemplo de saída 3**
6 000000 000010 000001 000000 010000 001000 2 4 4 5 1 2 4 4 6 1 2	S N

Hotel Nlogônia

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

José Enelogeu sempre teve o sonho de conhecer a Nlogônia, a cidade de origem de seus avós. Enelogeu acaba de vencer um concurso da emissora de televisão local cujo prêmio é uma viagem de D dias para a Nlogônia com tudo pago. Ele ficará hospedado no mais famoso hotel da cidade, o *Hotel Nlogônia*.

As férias de Enelogeu duram N dias. Ele pretende escolher alguns destes N dias para fazer a viagem à Nlogônia e se hospedar no famoso hotel, onde ele deseja ficar o máximo de dias possível. Como o prêmio só cobre D diárias do hotel, Enelogeu separou W *dólares nlogonianos* para gastar com diárias adicionais. Deste modo, a viagem de Enelogeu segue as seguintes regras:

- Todos os dias em que Enelogeu estará hospedado no hotel devem ser **consecutivos** (ou seja, ele fará uma única viagem à Nlogônia).
- Os D dias com hospedagem paga pelo concurso devem ser **consecutivos**. Enelogeu não gastará nada nestes D dias.
- Nos outros dias (antes ou depois dos D dias pagos pelo prêmio) em que estará hospedado no hotel, Enelogeu deve pagar o custo da diária do hotel para aquele dia.
- Enelogeu pode gastar no máximo W dólares nlogonianos com diárias do hotel.

Ajude Enelogeu a aproveitar suas férias: dados os valores das diárias do hotel em cada um dos N dias de férias de Enelogeu, o número D de diárias pagas pelo concurso e a quantidade W de dólares nlogonianos que Enelogeu pode gastar com hospedagem, determine o número máximo de dias que ele pode ficar hospedado no Hotel Nlogônia.

Entrada

A primeira linha da entrada contém três inteiros N, D e W indicando, respectivamente, o número de dias das férias de Enelogeu, o número de dias de hospedagem pagos pelo concurso e a quantidade de dólares nlogonianos que Enelogeu separou para gastar com hospedagem.

A segunda e última linha da entrada contém N inteiros representando a tabela de preços do Hotel Nlogônia. O i-ésimo desses inteiros, p_i, é o preço em doláres nlogonianos da diária do hotel no i-ésimo dia de férias de Enelogeu.

Saída

Seu programa deverá produzir uma única linha contendo somente um inteiro, o número máximo de dias em que Enelogeu consegue se hospedar no Hotel Nlogônia durante suas férias.

Restrições

- $1 \leq N \leq 200\,000$
- $1 \leq D \leq N$
- $1 \leq W \leq 1\,000\,000\,000$
- $1 \leq p_i \leq 1\,000\,000\,000$ para $1 \leq i \leq N$

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 (22 pontos):**
  - $N \leq 400$
  - $W \leq 3\,000$
- **Subtarefa 3 (11 pontos):**
  - $N \leq 3\,000$
  - $W \leq 3\,000$
  - $p_i = 1$ ou $p_i = W + 1$ para todo $1 \leq i \leq N$
- **Subtarefa 4 (23 pontos):**
  - $N \leq 3\,000$
  - $W \leq 3\,000$
- **Subtarefa 5 (12 pontos):**
  - $p_i \geq p_{i+1}$ para todo $1 \leq i < N$
- **Subtarefa 6 (32 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
`6 2 100` `65 50 50 60 45 60`	`4`

Explicação do exemplo 1: Neste caso, as férias duram 6 dias, o concurso paga por 2 dias e Enelogeu possui 100 dólares nlogonianos para gastar com hospedagem. As diárias no hotel custam, do primeiro ao sexto dia, 65, 50, 50, 60, 45 e 60 dólares nlogonianos.

Enelogeu consegue se hospedar no hotel por 4 dias se escolher os dias de 2 a 5, inclusive. Ele possui duas opções válidas:

- O concurso paga os dias 3 e 4 enquanto que Enelogeu paga os dias 2 e 5, gastando um total de $50 + 45 = 95$ dólares nlogonianos.
- O concurso paga os dias 4 e 5 enquanto que Enelogeu paga os dias 2 e 3, gastando um total de $50 + 50 = 100$ dólares nlogonianos.

Observe que Enelogeu não pode escolher pagar os dias 4 e 5 pois o custo de $60 + 45 = 105$ dólares nlogonianos excede o total de 100 dólares nlogonianos que ele possui.
É possível verificar que Enelogeu não consegue se hospedar no hotel por 5 ou mais dias.

Exemplo de entrada 2	**Exemplo de saída 2**
`5 3 60` `30 40 30 40 30`	`5`

Explicação do exemplo 2: Neste caso, Enelogeu consegue passar todos os 5 dias no hotel: ele pode pagar os dias 1 e 5, gastando um total de $30 + 30 = 60$ dólares nlogonianos, e pedir para o concurso pagar os dias 2, 3 e 4.

Exemplo de entrada 3	**Exemplo de saída 3**
`14 3 4` `1 1 5 5 1 5 1 5 1 1 5 5 1 1`	`6`

Explicação do exemplo 3: Observe que, se Enelogeu escolhesse os dias 3, 4 e 6 para serem pagos pelo concurso, ele conseguiria pagar as diárias dos dias 1, 2, 5 e 7 para ficar no hotel todos os 7 primeiros dias. Porém, isso não é um plano válido pois os dias pagos pelo concurso (assim como todos os dias da viagem) devem ser consecutivos.

Exemplo de entrada 4	**Exemplo de saída 4**
`6 1 35000` `30000 25000 20000 15000 10000 5000`	`4`