Competidor(a): _____

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

Olimpíada Brasileira de Informática

OBI2024

Caderno de Tarefas

Modalidade Programação • Nível 1 • 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.

Musical

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

Paca é uma menina apaixonada por música, dança e outras formas de expressão cultural. Sua mais nova paixão é o aplicativo *Musical*, uma plataforma de *streaming* de músicas que possui também o propósito de incentivar os usuários a entenderem mais sobre música por meio da criação de *listas de reprodução* (ou *playlists*).

Uma lista de reprodução no *Musical* é uma sequência de músicas que são reproduzidas (ou seja, tocadas) pelo aplicativo na ordem escolhida pelo usuário. Assim, a primeira música da lista é a primeira a ser tocada, a segunda música é a segunda a ser tocada, e assim por diante. As listas de reprodução usam sempre a opção de *loop*, o que significa que, após o término da última música, a reprodução recomeça, na mesma ordem, a partir da primeira música.

O aplicativo mostra informações sobre diversas características das músicas. Uma das características que mais interessam a Paca é a *energia* de uma música, que é indicada por um número inteiro entre 0 e 100 representando o quão animada e dançante é a música. Desta forma, músicas com energia 0 são muito calmas, enquanto que músicas com energia 100 são muito agitadas.

A nova funcionalidade do *Musical* é o *Medidor de Dissonância*, que mede o quanto a energia da música sendo tocada muda durante a reprodução de uma lista em *loop*. A *dissonância* de uma lista de reprodução é definida pelo aplicativo como a soma das variações de energia (ou seja, o valor absoluto em quanto a energia mudou) entre cada música e a música seguinte. Observe que a música seguinte à última música é sempre a primeira música.

Paca deseja criar uma lista de reprodução contendo N músicas que ela escolheu. Ela decidiu que cada música deve aparecer na lista exatamente uma vez, de modo que a lista possua exatamente N músicas. Agora, ela só precisa decidir em qual ordem as N músicas devem ser reproduzidas. Paca percebeu que, dependendo da ordem das músicas na lista, a dissonância pode ser maior ou menor. Por isso, ela pediu sua ajuda para criar uma lista que possua a menor dissonância possível.

Dadas as energias das N músicas escolhidas por Paca, descubra qual a menor dissonância possível em uma lista de reprodução contendo as N músicas, exatamente uma vez cada, em alguma ordem.

Entrada

A primeira linha da entrada contém um único inteiro N, representando a quantidade de músicas que Paca escolheu para a sua lista de reprodução.

As próximas N linhas descrevem as energias das músicas que Paca escolheu, em ordem arbitrária. A i-ésima destas linhas contém um único inteiro e_i, a energia da i-ésima música.

Saída

Seu programa deverá produzir uma única linha contendo somente um inteiro, a dissonância mínima em uma lista de reprodução contendo exatamente as N músicas escolhidas por Paca.

Restrições

- $3 \leq N \leq 10\,000$
- $0 \leq e_i \leq 100$ 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 (14 pontos):** $N = 3$, ou seja, a lista possui exatamente três músicas.
- **Subtarefa 3 (24 pontos):** $N \leq 8$, ou seja, a lista possui no máximo oito músicas.
- **Subtarefa 4 (13 pontos):** $e_i \leq 1$ para todo $1 \leq i \leq N$, ou seja, toda música possui energia 0 ou 1.
- **Subtarefa 5 (16 pontos):** $e_i \leq 2$ para todo $1 \leq i \leq N$, ou seja, toda música possui energia 0, 1 ou 2.
- **Subtarefa 6 (33 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
5 60 15 20 80 20	130

Explicação do exemplo 1: Neste exemplo, a lista de reprodução deve conter cinco músicas com energias 60, 15, 20, 80 e 20. Se as músicas estiverem nesta mesma ordem na lista, a variação de energia entre a primeira e a segunda música é $60 - 15 = 45$, enquanto que a variação de energia entre a segunda e a terceira música é $20 - 15 = 5$, a variação entre a quinta e a primeira música é $60 - 20 = 40$, e assim por diante. Deste modo, a dissonância da lista é

$$|60 - 15| + |15 - 20| + |20 - 80| + |80 - 20| + |20 - 60| = 210.$$

Porém, se a ordem das músicas na lista de reprodução fosse 20, 15, 20, 80 e 60, a dissonância da lista seria

$$|20 - 15| + |15 - 20| + |20 - 80| + |80 - 60| + |60 - 20| = 130.$$

É possível checar que não existe nenhuma lista que contém exatamente essas cinco músicas em alguma ordem e cuja dissonância é menor ou igual a 129. Assim, 130 é a menor dissonância possível.

Exemplo de entrada 2	**Exemplo de saída 2**
3 30 20 95	150

Exemplo de entrada 3	**Exemplo de saída 3**
10 0 0 1 1 0 1 0 1 0 0	2

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`

Brigadeiros

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

Você está na festa de formatura da escola e está quase na hora de servirem um dos doces favoritos dos brasileiros: brigadeiro.

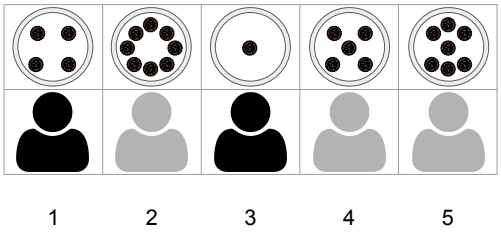
N alunos estão participando da festa e cada um vai comer um dos N pratos com brigadeiros que estão alinhados em cima da mesa do refeitório. Os pratos são numerados de 1 a N da esquerda para a direita. Os alunos estão sentados em frente à mesa na mesma ordem dos brigadeiros e cada um irá comer o prato de brigadeiros à sua frente. Cada prato possui entre 0 e 9 brigadeiros.

O seu grupo de amigos é formado por um subconjunto dos alunos e possui K membros (incluindo você). Seu grupo deseja comer muitos brigadeiros e desenvolveu um plano para que os membros do grupo mudem seus lugares de modo a maximizar a quantidade de brigadeiros que o grupo irá comer no total (ou seja, somando as quantidades que cada membro do grupo irá comer).

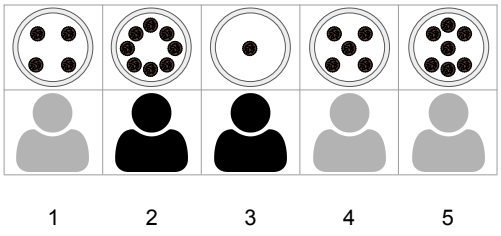
Cada membro do grupo pode pedir para trocar de lugar com um dos alunos sentados nas cadeiras vizinhas à dele (ou seja, os alunos sentados imediatamente à esquerda e à direita dele, se existirem). Os outros alunos não estão prestando atenção nos pratos e, por isso, sempre aceitam as trocas pedidas. Assim, um membro do grupo pode se mover para a esquerda ou para a direita usando várias trocas, sempre com um de seus vizinhos atuais.

Porém, para evitar que os outros alunos descubram o plano, vocês decidiram que somente uma troca pode ser realizada a cada segundo. Vocês sabem que faltam T segundos para que os brigadeiros sejam servidos, e portanto vocês podem fazer no máximo T trocas entre alunos vizinhos.

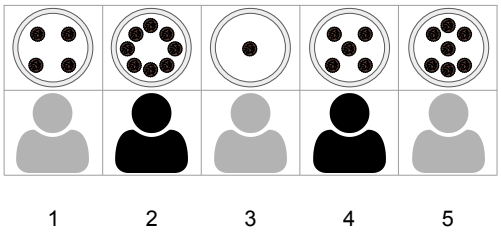
O diagrama abaixo ilustra um exemplo para $N = 5$, $K = 2$ e $T = 2$. Existem 5 pratos com 4, 8, 1, 5 e 7 brigadeiros, da esquerda para a direita. o grupo possui dois amigos que estão inicialmente nas posições 1 e 3, e faltam 2 segundos para os brigadeiros serem servidos.



No primeiro segundo, o membro na posição 1 pode trocar de lugar com o aluno na posição 2:



No último segundo, o membro na posição 3 pode trocar de lugar com o aluno na posição 4:



Desta forma, ao final dos dois segundos, os amigos do grupo estarão nas posições 2 e 4, e portanto irão comer 8 e 5 brigadeiros, respectivamente, totalizando $8+5 = 13$ brigadeiros. É possível verificar que esta é a quantidade máxima de brigadeiros que o grupo pode comer de acordo com as condições do exemplo.

Dados o número de alunos na festa (que também corresponde ao número de pratos), o número de amigos em seu grupo, o número de segundos que faltam para os brigadeiros serem servidos, quantos brigadeiros cada prato possui, e as posições iniciais dos membros do grupo, sua tarefa é determinar o máximo número de brigadeiros que seu grupo consegue comer no total. Observe que pode ser ótimo gastar alguns segundos sem realizar nenhuma troca.

Entrada

A primeira linha da entrada contém três inteiros N, K e T, a quantidade de pratos de brigadeiros, a quantidade de amigos no grupo e quantos segundos restam para os brigadeiros serem servidos, respectivamente.

A segunda linha representa os pratos de brigadeiros e contém N inteiros separados por espaços em branco. O i-ésimo destes inteiros é P_i, a quantidade de brigadeiros no i-ésimo prato.

A terceira e última linha representa as posições iniciais dos alunos na mesa e contém N inteiros separados por espaços em branco. O i-ésimo destes inteiros, G_i, indica se o aluno sentado na i-ésima cadeira é membro do grupo ou não: $G_i = 1$ se o aluno é membro do grupo, ou $G_i = 0$ caso contrário. É garantido que existem exatamente K valores iguais a 1 nesta linha.

Saída

Seu programa deverá produzir uma única linha contendo somente um inteiro, a quantidade máxima de brigadeiros que seu grupo consegue comer.

Restrições

- $1 \leq N \leq 300$
- $0 \leq P_i \leq 9$ para todo $1 \leq i \leq N$
- $1 \leq K \leq N$
- $0 \leq T \leq 1\,000\,000\,000$
- $G_i = 0$ ou $G_i = 1$ para todo $1 \leq i \leq N$
- Existem exatamente K índices i para os quais $G_i = 1$

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 (13 pontos):**
  - $N \leq 50$
  - $K = 3$
  - $T \leq 1\,000$
- **Subtarefa 3 (22 pontos):**
  - $N \leq 16$
  - $T \leq 1\,000$
- **Subtarefa 4 (23 pontos):**
  - $N \leq 50$
  - $T \leq 1\,000$
- **Subtarefa 5 (10 pontos):**
  - $N \leq 50$
  - $T \leq 100\,000$
- **Subtarefa 6 (11 pontos):**
  - $N \leq 100$
- **Subtarefa 7 (21 pontos):** Sem restrições adicionais.

Exemplos

Exemplo de entrada 1	**Exemplo de saída 1**
5 2 2 4 8 1 5 7 1 0 1 0 0	13

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

Exemplo de entrada 2	**Exemplo de saída 2**
4 2 3 8 9 1 5 0 1 0 1	17

Exemplo de entrada 3	**Exemplo de saída 3**
4 2 2 8 9 1 5 0 1 0 1	14

Explicação do exemplo 3: Neste exemplo, os amigos não tem tempo suficiente para conseguir comer 17 brigadeiros (como no exemplo 2). A melhor opção é que eles continuem em suas posições iniciais e comam $9 + 5 = 14$ brigadeiros.

Exemplo de entrada 4	**Exemplo de saída 4**
15 7 100 7 3 0 8 6 1 9 1 5 8 1 6 3 4 9 1 1 0 0 1 0 1 0 0 0 1 0 1 1 0	53