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 (23 de agosto de 2023).

OBI

Olimpíada Brasileira de Informática

OBI2023

Caderno de Tarefas

Modalidade Programação • Nível Júnior • Fase 2 - Turno B

23 de agosto de 2023

A PROVA TEM DURAÇÃO DE 2 HORAS

Promoção:

SBC Sociedade Brasileira de Computação

Apoio:

FUNDAÇÃO BEHRING

Coordenação:

INSTITUTO DE COMPUTAÇÃO UNICAMP

Instruções

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

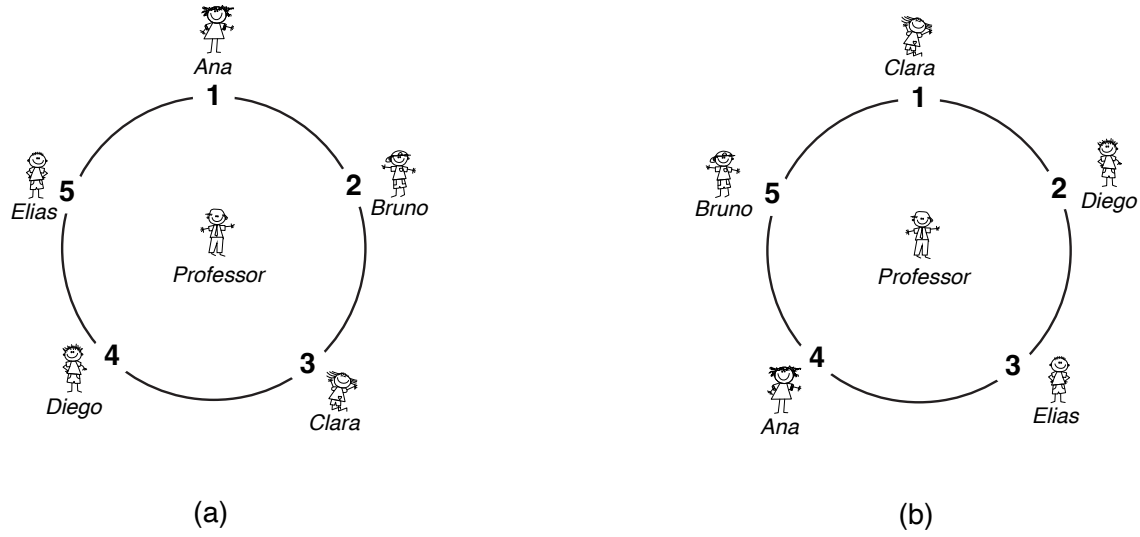
- Este caderno de tarefas é composto por 8 páginas (não contando a folha de rosto), numeradas de 1 a 8. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta 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 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).
- Você pode submeter até 50 soluções para cada tarefa. A pontuação total de cada tarefa é a melhor pontuação entre todas as submissões. Se a tarefa tem sub-tarefas, para cada sub-tarefa é considerada a melhor pontuação entre todas as submissões.
- 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 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.

Brincadeira de Roda

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

Bruno convidou seu professor e colegas de classe para uma conferência da OBI (*Organização de Brincadeiras Infantis*). Hoje, o tema da conferência é "Brincadeiras de Roda." As brincadeiras do dia seguem todas a mesma estrutura: Bruno e seus colegas formam uma roda enquanto o professor, que não faz parte da roda, fica no centro dela. O professor aponta para a posição de um dos alunos, indicando que aquela posição é representada pelo número 1. As outras posições são então numeradas no sentido horário, em ordem crescente. A figura (a) indica um possível cenário inicial da brincadeira para uma classe com 5 alunos.

Quando a brincadeira começa, os alunos devem ficar em suas posições (participando de outras dinâmicas) até o professor bater palmas: toda vez que ouvem palmas, todos os alunos se movem uma posição no sentido horário (ou seja, o aluno na posição 1 vai para a posição 2, o aluno na posição 2 vai para a posição 3, e assim por diante; o aluno na última posição se move para a posição 1). A figura (b) indica como fica a roda da figura (a) após o professor bater palmas três vezes.



Bruno e alguns outros alunos foram beber água, enquanto a brincadeira continuou, com o professor batendo palmas algumas vezes. Bruno agora quer voltar para sua posição correta (ou seja, sua posição se não tivesse saído para beber água). Ele lembra a posição em que estava logo antes de sair, e conseguiu ouvir todas as vezes que o professor bateu palmas enquanto ele estava fora. Agora, ele pede sua ajuda: dados o número de estudantes na classe, a posição inicial de Bruno, e o número de vezes que o professor bateu palmas enquanto ele foi beber água, ajude Bruno a descobrir para qual posição na roda ele deve voltar.

Entrada

A primeira linha de entrada contém um inteiro N indicando o número de alunos na classe, incluindo Bruno. As posições na roda são numeradas de 1 a N no sentido horário. A segunda linha contém um inteiro I, indicando a posição inicial de Bruno (ou seja, a posição dele logo antes de sair para beber água). A terceira e última linha contém um inteiro P, indicando o número de vezes que o professor bateu palmas enquanto Bruno estava fora da roda.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, a posição na roda para a qual Bruno deve voltar.

Restrições

- $3 \leq N \leq 100$
- $1 \leq I \leq N$
- $1 \leq P \leq 1\ 000$

Informações sobre a pontuação

A tarefa vale 100 pontos.

Exemplos

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

Explicação do exemplo 1: esse é o exemplo da figura (b) do enunciado. Temos 5 estudantes na roda e Bruno estava na posição 2. Após o professor bater palmas três vezes, Bruno deve ir para a posição 5.

Exemplo de entrada 2	**Exemplo de saída 2**
10 7 30	7

Explicação do exemplo 2: Depois de 30 rodadas de palmas, todos os 10 estudantes estão de volta em suas posições iniciais (posições em que estavam quando Bruno saiu para beber água).

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

UPA

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

A Unicamp organiza anualmente, sempre em agosto, a UPA – Unicamp Portas Abertas –, um evento em que alunos de escolas da região podem visitar e conhecer as atividades da Universidade. O evento recebe em um dia dezenas de milhares alunos, que chegam em ônibus.

Um dos grandes problemas da organização da UPA é o estacionamento dos ônibus que trazem os alunos. A Unicamp reserva uma longa avenida para que os ônibus estacionem, um em seguida ao outro, todos paralelos à avenida; para cada ônibus são reservados 20 metros de avenida, com os espaços marcados no chão.

Para organizar o estacionamento, cada escola que chega de ônibus deve informar o horário de chegada e o horário de partida de seu ônibus, e os horários devem ser obedecidos rigorosamente. Dessa forma, o espaço de um ônibus que já partiu pode ser reutilizado por um outro ônibus que chega.

Dados os horários em que cada ônibus chega e parte, escreva um programa para calcular o menor espaço possível, em metros, que deve ser reservado para que todos os ônibus estacionem.

Entrada

A primeira linha contém um inteiro N, o número de ônibus que chegarão para a UPA. Os horários de chegada e partida são dados como números inteiros. Cada uma das N linhas seguintes descreve um ônibus e contém dois inteiros C_i e P_i, respectivamente o instante de chegada e o instante de partida do ônibus i. Dois ônibus A e B tais que $C_B \geq P_A$ ou $C_A \geq P_B$ podem utilizar o mesmo espaço de estacionamento. Assim, por exemplo, se o horário de chegada do ônibus B e a partida do ônibus A forem iguais, então o ônibus B pode ocupar o espaço que o ônibus A ocupou.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o menor comprimento da avenida, em metros, que deve ser reservado para o estacionamento de ônibus.

Restrições

- $1 \leq N \leq 100\ 000$
- $0 \leq C_i \leq 100\ 000$ para $1 \leq i \leq N$
- $0 \leq P_i \leq 100\ 000$ para $1 \leq i \leq N$
- $C_i < P_i$ para $1 \leq i \leq N$

Informações sobre a pontuação

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

- **Subtarefa 1 (12 pontos):** $P_i = 1 + C_i$ para $1 \leq i \leq N$

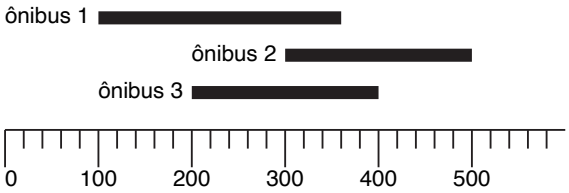
- **Subtarefa 2 (29 pontos):** $N \leq 1\,000$, $C_i \leq 1\,000$ e $P_i \leq 1\,000$.
- **Subtarefa 3 (27 pontos):** É garantido que a resposta é menor do que 1 000 metros.
- **Subtarefa 4 (32 pontos):** Nenhuma restrição adicional.

Seu programa pode resolver corretamente todas ou algumas das subtarefas acima *(elas não precisam ser resolvidas em ordem)*. Sua pontuação final na tarefa é a soma dos pontos de todas as subtarefas resolvidas corretamente por qualquer uma das suas submissões.

Exemplos

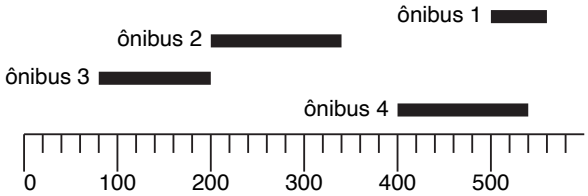
Exemplo de entrada 1	**Exemplo de saída 1**
3 100 360 300 500 200 400	60

Explicação do exemplo 1: há três ônibus. O primeiro chega no instante 100 e parte no instante 360, o segundo chega no instante 300 e parte no instante 500, e o terceiro chega no instante 200 e parte no instante 400. A figura abaixo ilustra o tempo em que cada ônibus utiliza o estacionamento. A resposta é 60 metros (espaço para 3 ônibus).



Exemplo de entrada 2	**Exemplo de saída 2**
4 500 560 200 340 80 200 400 540	40

Explicação do exemplo 2: há quatro ônibus. O primeiro chega no instante 500 e parte no instante 560, o segundo chega no instante 200 e parte no instante 340, o terceiro chega no instante 80 e parte no instante 200, e o quarto chega no instante 400 e parte no instante 540. A figura abaixo ilustra o tempo em que cada ônibus utiliza o estacionamento. A resposta é 40 metros (espaço para 2 ônibus) – por exemplo, os ônibus 1 e 2 podem usar um dos espaços e os ônibus 3 e 4 podem usar o outro.



Empresa

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

Devido ao seu sucesso na primeira fase da OBI, a pequena empresa IOI (*Insatisfação Observada International*) te ofereceu uma entrevista de estágio. Como parte do processo seletivo, você deve ajudar a empresa a avaliar a satisfação dos funcionários com seus empregos.

Na hierarquia da organização, cada funcionário (exceto o fundador) possui um chefe. A empresa valoriza experiência no trabalho: o chefe de todo funcionário entrou na empresa antes dele. Para cada funcionário, os seus *subordinados diretos* são os funcionários que têm ele como chefe.

Cada funcionário possui um salário mensal, e um estudo feito pela empresa revelou que existe um jeito simples de descobrir quais funcionários estão insatisfeitos: um funcionário está insatisfeito se, e somente se, algum dos seus subordinados **diretos** ganha mais do que ele (note que ele está satisfeito se seus subordinados ganham tanto quanto ele).

Com o fim do trimestre chegando, o time de Recursos Humanos está considerando ajustar os salários de alguns funcionários, possivelmente aumentando ou reduzindo o valor mensal (porém, para evitar confusão, eles podem alterar o salário de cada funcionário no máximo uma vez). Mas, primeiro, eles precisam avaliar a insatisfação geral dos funcionários durante essa série de ajustes.

Sua tarefa é, dada a estrutura hierárquica da empresa, os salários iniciais, e a sequência considerada de ajustes, determinar o número total de funcionários insatisfeitos inicialmente e após cada ajuste.

Entrada

A primeira linha de entrada contém um inteiro N, indicando o número de funcionários da empresa. Os funcionários são identificados de 1 a N de acordo com a ordem em que entraram na organização. O fundador é identificado pelo número 1.

Cada uma das N linhas seguintes descreve um funcionário: a linha $i + 1$ contém dois inteiros, C_i e S_i, indicando o chefe e o salário do funcionário i, respectivamente. É garantido que $C_1 = 1$ (consideramos que o fundador é chefe de si mesmo) e que $C_i < i$ para todos os outros funcionários (ou seja, o chefe de um funcionário entrou antes dele).

A próxima linha contém um inteiro A, indicando o número de ajustes de salários a serem feitos. Finalmente, as próximas A linhas contêm a sequência de ajustes: cada linha contém dois inteiros, I e X, indicando que o salário mensal do funcionário I foi alterado para X. É garantido que nenhum funcionário tem o salário alterado mais do que uma vez. Observe que os ajustes são cumulativos (**não** desfazemos o ajuste $i - 1$ antes de fazer o ajuste i).

Saída

Seu programa deve produzir $A + 1$ linhas, cada uma contendo um único inteiro. A primeira linha deve conter o número de funcionários insatisfeitos antes dos ajustes. As próximas A linhas devem conter os números de funcionários insatisfeitos após cada ajuste.

Restrições

- $1 \leq N \leq 2\ 000$
- $0 \leq A \leq N$
- $C_1 = 1$ e $1 \leq C_i < i$ para $i > 1$
- $1 \leq S_i \leq 1\ 000\ 000$

- $1 \leq I \leq N$
- $1 \leq X \leq 1\ 000\ 000$
- Os valores de I são todos distintos entre si (ou seja, cada funcionário tem o salário ajustado no máximo uma vez).

Informações sobre a pontuação

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

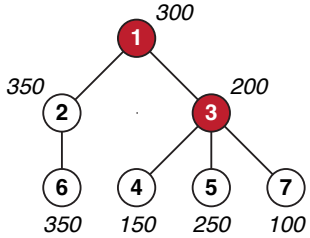
- **Subtarefa 1 (40 pontos):** $N \leq 100$ e $A = 0$ – ou seja, nenhum ajuste será feito, e você só precisa calcular a insatisfação total inicial. *(Veja o exemplo de entrada 1.)*
- **Subtarefa 2 (30 pontos):** $N \leq 100$.
- **Subtarefa 3 (30 pontos):** Nenhuma restrição adicional.

Seu programa pode resolver corretamente todas ou algumas das subtarefas acima *(elas não precisam ser resolvidas em ordem)*. Sua pontuação final na tarefa é a soma dos pontos de todas as subtarefas resolvidas corretamente por qualquer uma das suas submissões.

Exemplos

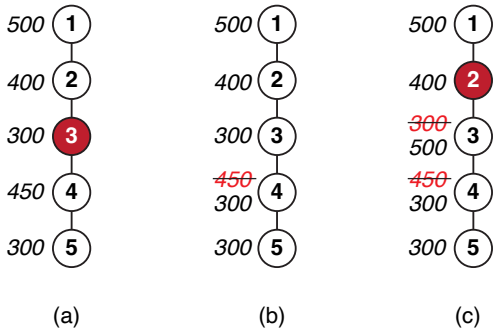
Exemplo de entrada 1	**Exemplo de saída 1**
7 1 300 1 350 1 200 3 150 3 250 2 350 3 100 0	2

Explicação do exemplo 1: Veja a figura abaixo. Os funcionários insatisfeitos são o 1 (pois seu subordinado direto 2 tem um salário maior) e o 3 (pois seu subordinado direto 5 tem um salário maior). Note que o funcionário 2 não está insatisfeito de seu subordinado direto 6 ganhar exatamente o mesmo que ele.



Exemplo de entrada 2	**Exemplo de saída 2**
5 1 500 1 400 2 300 3 450 4 300 2 4 300 3 500	1 0 1

Explicação do exemplo 2: Inicialmente (figura a), apenas o funcionário 3 está insatisfeito. Observe que, apesar do funcionário 4 ter um salário maior que o funcionário 2, o funcionário 2 não está insatisfeito pois o 4 não é um subordinado direto do 2. Após o primeiro ajuste (figura b), o salário do funcionário 4 cai, deixando o funcionário 3 satisfeito. Após o segundo ajuste (figura c), o salário do funcionário 3 sobe, tornando o funcionário 2 insatisfeito.



Exemplo de entrada 3	**Exemplo de saída 3**
6 1 100 1 200 1 300 2 400 3 500 3 600 3 1 500 3 550 4 200	3 2 3 2