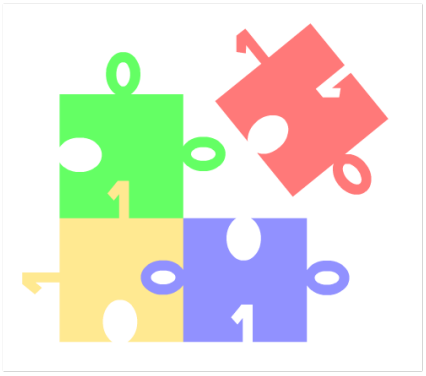


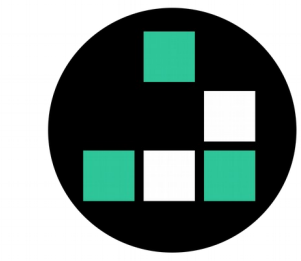
8ª Mini Maratona de Programação UFSCar Sorocaba 2018

24 de agosto de 2018

Sessão Principal

Este caderno de problemas contém 7 problemas em páginas numeradas de 1 a 12.

Apoio:



Informações Gerais

Exceto quando explicitamente indicado, as seguintes condições valem para todos os problemas:

Entrada

- A entrada deve ser lida da entrada padrão.
- A entrada é composta de um único caso de teste, descrito em um número de linhas que depende do problema.
- Quando uma linha de dados contém vários valores, estes são separados por um único espaço em branco. Não aparecem outros espaços na entrada. Não existem linhas vazias.
- Cada linha, incluindo a última, possui exatamente um caractere de final-de-linha.
- O fim da entrada coincide com o final do arquivo.

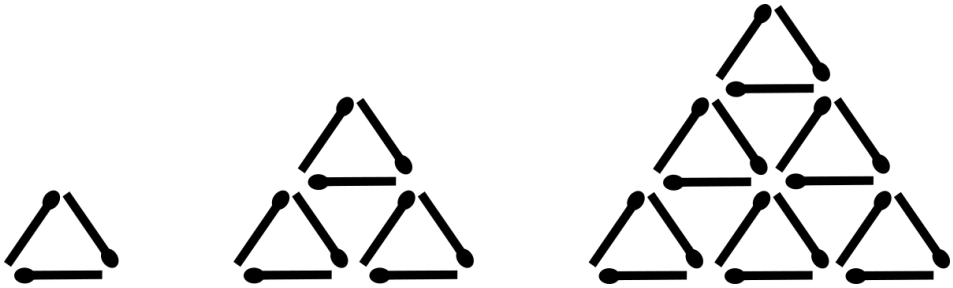
Saída

- A saída deve ser escrita na saída padrão.
- O resultado do caso de teste deve aparecer na saída usando um número de linhas que depende do problema.
- Quando uma linha de saída contém vários valores, estes devem ser separados por um único espaço. Não podem aparecer outros espaços na saída. Não devem existir linhas vazias.
- Cada linha, incluindo a última, deve possuir exatamente um caractere de final-de-linha.
- Nenhum indicador especial deve ser escrito para indicar o fim da saída.
- O formato da saída deve seguir rigorosamente o padrão descrito no problema.

Problema A

Palitos

Nome do arquivo: palitos.[c | cpp | java]



Quem nunca brincou com uma caixa de palitos? Alicinha adora montar pirâmides de triângulos com eles, seguindo os padrões formados acima. Cada vez que monta uma pirâmide, ela se dedica a montar uma maior. O problema é determinar o número de palitos que precisa reservar para o seu próximo projeto.

Ajude a Alicinha a saber quantos palitos irá precisar para montar a sua nova pirâmide.

Entrada

A entrada contém uma única linha com um único número inteiro B representando o número de palitos utilizados na base da pirâmide ($1 \leqslant B \leqslant 10^8$).

Saída

A saída consiste em uma única linha, indicando o número total de palitos necessários para montar toda a pirâmide.

Exemplos

Entrada	**Saída**
3	18

Entrada	**Saída**
4	30

Problema B

Orquídeas

Nome do arquivo: orquideas.[c | cpp | java]

A empresa DComp-So (Distribuição e Comércio de Orquídeas Multicoloridas Phalaenopsis de Sorocaba) possui diversas estufas para o cultivo de orquídeas espalhadas pela região de Sorocaba. Para cada cor de orquídea Phalaenopsis certas condições ambientais são mais favoráveis para o cultivo de belas pencas de orquídeas.

Existem ligações dos locais das estufas para uma rede de pontos de distribuição das orquídeas da empresa para todos os estabelecimentos quem vendem suas orquídeas para o consumidor final. De toda estufa é possível chegar em qualquer centro de distribuição, embora possa ser necessário passar por outros centros de distribuição no caminho. Cada uma das ligações possui um custo de transporte que é proporcional à distância e outros fatores logísticos.

Os negócios da empresa estão indo de vento em popa ultimamente, e o volume das vendas aumentou muito em pouco tempo. Surgiu então a necessidade de otimizar o custo das entregas das orquídeas ao centros de distribuição. Para isso, o gerente de vendas busca saber para cada centro de distribuição qual a estufa ideal para fazer o abastecimento, levando em consideração o custo de transporte para se utilizar cada uma das ligações disponíveis.

Entrada

A primeira linha da entrada contém três inteiros N, M e K. O valor N ($1 \leqslant N \leqslant 10^4$) representa quantos são no total as estufas e os pontos de distribuição. Cada um desses pontos é representado por um inteiro entre 0 e N - 1. As estufas são representadas pelos K menores números e os centros de distribuição pelos demais, isto é, os números K, ..., N - 1.

O valor M ($1 \leqslant M \leqslant 10^5$) representa o número total de ligações entre estufas e centros de distribuição e entre os centros de distribuição. As M linhas subsequentes contêm pares de inteiros A e B ($0 \leqslant A,B \leqslant N - 1$) e um número C ($1 \leqslant C \leqslant 10^4$) que representa o custo de transporte por essa ligação.

Saída

A saída consiste em uma única linha, contendo N - K inteiros, representando os números das estufas que atendem com menor custo cada um dos centros de distribuição. Os N - K valores devem ser apresentados em ordem crescente do número do centro de distribuição, isto é, na ordem K, ..., N – 1. Caso mais de uma estufa atenda um centro de distribuição com o mesmo custo, deve ser escolhida a estufa com menor número.

Exemplos

Entrada	**Saída**
8 10 3 0 6 35 1 3 15 2 4 12 3 4 23 3 5 25 3 6 34 3 7 10 4 5 10 4 7 5 5 6 15	1 2 2 0 2

Entrada	**Saída**
6 8 2 0 2 20 0 3 15 1 5 20 2 1 6 2 4 17 3 4 10 4 1 22 4 5 14	1 0 1 1

Problema C

Montanhas Quadráticas

Nome do arquivo: montanhas.[c | cpp | java]

Logginho adora desenhar montanhas! Este ano, Logginho finalmente aprendeu equações de segundo grau na escola, e percebeu uma coisa muito interessante sobre estas equações: Dependendo dos coeficientes escolhidos, o gráfico da equação acima do eixo x se parece com uma montanha!

Agora, Logginho está obcecado por suas “montanhas quadráticas” e não para de desenhá-las! Porém, um problema surgiu e Logginho acredita que só você é capaz de resolvê-lo: Dependendo dos coeficientes escolhidos, seus desenhos ficam estranhos, e não parecem montanhas! Para Logginho, uma montanha legal é uma montanha que possui área positiva, mas não infinita. Sua tarefa é determinar, para uma montanha dada pela fórmula $y = ax^2 + bx + c$, se esta montanha é legal ou não.

Entrada

A entrada é dada por única linha contendo três números reais A, B e C, representando respectivamente os coeficientes da equação de segundo grau ($-100 \leqslant A, B, C \leqslant 100$).

Saída

A saída deve conter uma única linha com o valor -1 se a montanha não for legal, ou a área da montanha caso ela seja legal. Arredonde sua resposta para duas casas decimais.

Exemplos

Entrada	**Saída**
1 2 3	-1

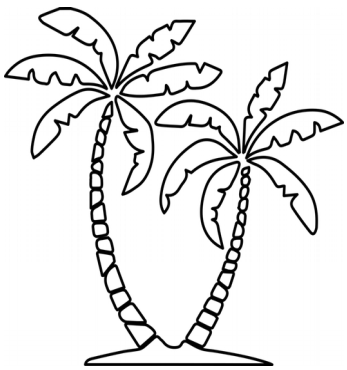
Entrada	**Saída**
-5 30 10	243.22

Entrada	**Saída**
-5 -7 -8	-1

Problema D

Caribe

Nome do arquivo: caribe.[c | cpp | java]



Todos os anos, a empresa *Beterrabas Jr.* distribui prêmios para os seus funcionários. O presidente da empresa, Sr. Carlos, decidiu que neste ano levará seus funcionários para o Caribe. Carlos gosta muito das praias paradisíacas do Caribe. Como não será possível premiar todos os funcionários da empresa, será necessário selecionar quais funcionários irão viajar.

Pensando no bem-estar de todos, o presidente pediu que a escolha dos premiados considere que em toda relação de chefia, o chefe ou o subordinado sejam obrigatoriamente premiados. Assim, todos terão algum colega escolhido, ou será um dos premiados.

Carlos é o presidente da empresa, ou seja, não possui chefe imediato. Os demais funcionários possuem apenas uma chefia imediata, e podem ter ou não subordinados.

Ajude esta empresa júnior a encontrar o menor número de prêmios para financiar.

Entrada

A primeira linha da entrada contém um inteiro R ($1 \leqslant R \leqslant 500$), referente ao número de relações de chefia imediata. Em seguida estão R linhas, onde cada linha contém a identificação do(a) chefe C, seguida pela identificação do(a) subordinado(a) S, separados por um espaço. Tanto a identificação do chefe quanto a dos subordinados possui no mínimo 1 e no máximo 20 caracteres. O presidente da empresa é identificado como “Carlos”.

Saída

A saída consiste em uma única linha, indicando o menor número de funcionários que devem ser premiados.

Exemplos

Entrada	**Saída**
2 Carlos Serafina Serafina Joao	1

Entrada	**Saída**
2 Carlos Serafina Carlos Joao	1

Entrada	**Saída**
3 Carlos Serafina Carlos Joao Serafina Andre	2

Entrada	**Saída**
6 Andre Joao Carlos Marcos Marcos Paulo Paulo Sonia Andre Paula Carlos Andre	3

Problema E

Espionagem

Nome do arquivo: espionagem.[c | cpp | java]

Olá! Sou diretor da empresa BCC - *Business Competitiveness Consulting* e preciso de serviços especializados para uma investigação. Descobrimos que uma falha de configuração no algoritmo de criptografia utilizado por um de nossos concorrentes permite a recuperação de dados criptografados com determinada chave. Fomos informados que, conhecendo um par (T,C) de texto claro (original) e seu respectivo texto cifrado com uma chave K, ou parte destes, é possível recuperar outros textos cifrados com a mesma chave, mesmo sem descobrir a chave utilizada. Conseguimos obter um conjunto de mensagens de comunicação cifradas com diferentes chaves. O trabalho consiste em descobrir quais mensagens foram cifradas com a mesma chave de um par de texto conhecido (T,C) e decifrá-las. Fomos informados que vocês seriam OS especialistas para este trabalho.

Nossas fontes nos informaram que a criptografia é realizada com cifrador de fluxo (*stream cipher*), que utiliza uma chave do tamanho da mensagem e a cifragem corresponde apenas à operação OU EXCLUSIVO bit a bit entre a chave e o texto claro (C = T XOR K). Este tipo de cifrador tem como requisito de segurança nunca usar a mesma chave para duas cifragens.

As mensagens capturadas são de 4 bytes, estruturadas de acordo com a seguinte tabela (o bit mais significativo está a esquerda):

Bits	4	4	4	4	16
Conteúdo	Agência origem	Conta origem	Agência destino	Conta destino	Valor

A falha de segurança que descobrimos é que o sistema do concorrente utiliza a mesma chave sempre que a mensagem possui mesma conta de origem, ou seja, mesmo byte mais significativo. Pior ainda, a chave é selecionada de forma que operações com conta de origem diferentes sempre terão alguma diferença no primeiro byte do texto cifrado.

Entrada

A entrada é composta por números inteiros de 4 bytes sem sinal, representando as mensagens estruturadas conforme a tabela acima. A primeira linha contém a mensagem conhecida em texto claro T. A segunda linha contém a mensagem T cifrada com a chave K, correspondendo ao texto cifrado C. A terceira linha contém o número N de mensagens cifradas capturadas ($1 \leqslant N \leqslant 10^8$), seguido da lista de N mensagens cifradas a serem avaliadas, uma por linha.

Saída

A saída deve conter a lista de valores das transações (campo valor das mensagens decifradas) que possuem a mesma agência e conta de origem da mensagem conhecida. Os valores devem ser listados um por linha, na ordem das mensagens recebidas. Caso não exista nenhuma transação com a mesma agência e conta de origem da mensagem conhecida, a saída deve conter uma única linha com o valor -1.

Exemplos

Entrada	**Saída**
287440897 270663681 1 270663688	8

Entrada	**Saída**
287440897 270663681 1 539099137	-1

Entrada	**Saída**
3665112892 332770132 2 334678365 317861692	19765

Problema F

Fila de Banco

Nome do arquivo: fila.[c | cpp | java]

Mesmo com toda a tecnologia atual, muitas pessoas ainda vão ao banco pagar contas e fazer outras operações. Na agência central do Banco da Nlogônia (BN), todo o dia às 10 da manhã, quando as portas do banco de abrem é a maior confusão. As pessoas se aglomeram na porta e entram, ao mesmo tempo e de forma desordenada, para aí então formar a fila. Atender primeiro quem chega primeiro parece justo, mas nesse situação acaba sendo atender primeiro quem usa os cotovelos mais, o que não é lá muito justo.

Pensando em resolver essa situação, analistas do banco chegaram a seguinte conclusão: a fila inicial não deve ser formada por quem chegou primeiro. As pessoas que entram na agência às 10h devem ser atendidas de forma a minimizar o tempo total que todas elas ficam na fila. Isso é o ideal para o grupo, mas pode parecer injusto para alguns clientes. Por exemplo, um cliente que tenha 6 contas a pagar deve ser atendido depois de um cliente que tenha 4 contas e de outro que tenha 3 contas. Ele esperará o equivalente ao pagamento de 7 contas, mas isso é melhor do atender esse cliente primeiro e obrigar todos os outros clientes a esperar o pagamento de 6 contas.

Após uma série de simulações o BN chegou a conclusão que minimizar o tempo global de espera na fila é a solução mais justa e decidiu implantá-la. Como esse banco é famoso por prezar pelo bem estar dos seus clientes e estando seguro da eficiência dessa nova forma de organizar a fila, decidiu dar pequeno incentivo econômico para as pessoas aceitarem a nova forma de construir fila. Cada cliente receberá R$1 por minuto que ficou na fila. Assim, os clientes com operações demoradas receberão um incentivo financeiro para abrir mão de seu lugar na fila.

Comovido com o uso da tecnologia para ajudar as pessoas e não para desumanizá-las, você aceitou ajudar o banco a implementar essa ideia. Sua missão é receber uma lista do número de operações que cada cliente deve realizar, organizar a fila de forma a minimizar o tempo total que as pessoas esperam na fila e calcular quanto será o valor total que o banco deve pagar como incentivo aos clientes. Suponha que o banco paga R$1 para cada minuto de espera de cada pessoa, que cada operação bancária leva 1 minuto, e que todos os clientes serão atendidos, um por vez, em uma ordem que minimize o tempo de espera total.

Entrada

A primeira linha contém um número inteiro N representando o número de pessoas na fila ($1 \leqslant N \leqslant 10^4$). A linha seguinte contém N inteiros O, representando o número de operações a serem realizadas por cada pessoa ($1 \leqslant O \leqslant 10$).

Saída

A saída consiste em uma única linha contendo o valor total a ser pago, que corresponde ao tempo de espera total em minutos para todos os clientes na fila.

Exemplos

Entrada	**Saída**
3 3 6 4	10

Entrada	**Saída**
5 7 3 6 3 1	25

Problema G

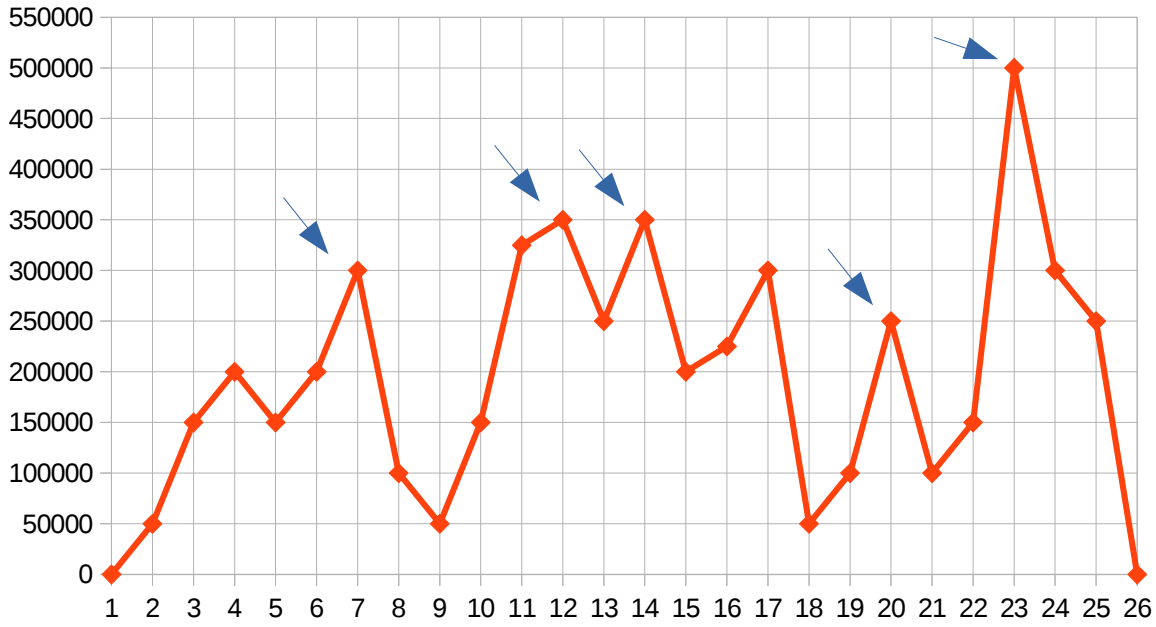
De Bolsa Para o Futuro

Nome do arquivo: bolsa.[c | cpp | java]

Para ganhar dinheiro na bolsa de valores é muito simples: compre quando o preço for baixo e venda quando o preço estiver alto. O problema é saber quando o preço não vai mais baixar ou subir, para saber o momento certo de negociar um papel. Bom, quer dizer, se você não tiver feito amizade com um viajante do tempo que te ofereceu de presente as cotações futuras de um papel amplamente negociado na bolsa.

Sabendo todos os valores de fechamento de negociação de ação é possível ganhar muito dinheiro. Porém, o seu amigo pediu para você não dar na cara que sabe como o valor das ações se comportará, então você pensou em uma estratégia para disfarçar o seu conhecimento e ainda ganhar muito dinheiro. Ao invés de comprar nos vales e vender nos picos, você vai comprar em dias aleatórios e vender apenas em dias de *super valorização*.

Um dia de super valorização é definido pelo seu *grau* de valorização. O grau de valorização v de um dia que termine com preço p é dado pelo maior v de forma que a evolução do preço para qualquer outro dia (antes ou depois do dia atual) que termine com um preço *estritamente maior* que p passará por um valor equivalente a p - v. Se o papel nunca fica mais caro que p, então v = p. Um dia de super valorização é um dia onde o grau de valorização v é maior ou igual a 150000 centavos. Na figura abaixo, os dias de super valorização são os dias 7, 12, 14, 20 e 23. O dia 17 não é um dia de super valorização pois do dia 17 para o dia 14 o preço caiu apenas 100000 centavos.



A sua tarefa é criar um programa para dados os valores da cotação de uma ação em dias consecutivos indicar quais desses dias são dias de super valorização.

Entrada

A primeira linha contém um número inteiro N representando o número de cotações a serem avaliadas ($3 \leqslant N \leqslant 10^5$). A linha seguinte contém N inteiros P, representando a cotação de fechamento da ação para cada um dos dias ($0 \leqslant P \leqslant 10^6$), sendo o índice do primeiro dia 1 e o índice do último dia N. Dias consecutivos sempre tem preços diferentes, e o primeiro e último pontos tem preço zero. Existe pelo menos um dia de super valorização na série de preços.

Saída

A saída deve conter uma única linha com os índices dos dias onde ocorrem super valorizações, em ordem crescente.

Exemplos

Entrada	**Saída**
5 0 10000 100000 884813 0	4

Entrada	**Saída**
7 0 100000 0 200000 180000 200000 0	4 6