Carteiro

Nome do arquivo fonte: `carteiro.c`, `carteiro.cpp`, `carteiro.pas`, `carteiro.java`, *ou* `carteiro.py`

Um carteiro é o responsável por entregar as encomendas na rua de Joãozinho. Por política da empresa, as encomendas devem ser entregues na mesma ordem que foram enviadas, mesmo que essa não seja a forma mais rápida. Cansado de subir e descer aquela rua tantas vezes, nosso amigo quer mostrar à empresa quanto tempo ele leva para entregar as encomendas, na tentativa de derrubar essa política.

A rua de Joãozinho tem N casas. Naturalmente, as casas são numeradas de forma ordenada (não necessariamente por números consecutivos). Como as casas possuem aproximadamente o mesmo tamanho, você pode assumir que o carteiro leva uma unidade de tempo para caminhar de uma casa até a casa imediatamente vizinha.

Há M encomendas para essa rua, que devem ser entregues na mesma ordem em que chegaram. Cada encomenda contém o número da casa onde deve ser entregue.

Escreva um programa que determine quanto tempo o carteiro levará para entregar todas as encomendas, assumindo que quando o tempo começa a contar, ele está na primeira casa (a de menor número), e o tempo termina de contar quando todas as encomendas foram entregues (mesmo que o carteiro não esteja de volta na primeira casa). Você pode desprezar o tempo para colocar a encomenda na caixa de correio (ou seja, se ele só tiver uma encomenda, para a primeira casa, a resposta para o problema é zero).

Entrada

A primeira linha contém dois inteiros, N e M, respectivamente o número de casas e o número de encomendas. A segunda linha contém N inteiros em ordem estritamente crescente, indicando os números das casas. A terceira linha contém M inteiros indicando os números das casas onde as encomendas devem ser entregues, na ordem dada na entrada.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o tempo que o carteiro levará para entregar todas as encomendas na ordem correta, assumindo que ele começa na casa de menor número.

Restrições

- $1 \leq N \leq 45.000$ e $1 \leq M \leq 45.000$
- O número de cada casa é um inteiro entre 1 e 1.000.000.000

Informações sobre a pontuação

- Para um subconjunto dos casos de teste totalizando 30 pontos, $1 \leq N \leq 1000$ e $1 \leq M \leq 1000$.

Exemplos

Entrada	**Saída**
5 5 1 5 10 20 40 10 20 10 40 1	10

Entrada	**Saída**
3 4 50 80 100 80 80 100 50	4

Ogros

Nome do arquivo fonte: `ogros.c`, `ogros.cpp` *ou* `ogros.pas`

Ogros marcianos, como todos sabem, são extremamente fortes. Numa feira de circo marciano, ogros são chamados para bater um martelo num aparelho que mede sua força. O ogro ganha um determinado prêmio dependendo da faixa de força que alcançou (por exemplo, se a força alcançada foi entre 0 e 5, ganha 10 pontos. Se for entre 6 e 10, ganha 30). São possíveis N premiações, para N faixas de força alcançadas.

Você deve escrever um programa que recebe quais são as faixas de forças e suas respectivas premiações. Em seguida, o programa recebe a força alcançada por M ogros, e deve calcular quanto cada ogro recebeu de premiação.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado). A primeira linha contém dois inteiros N e M ($2 \leq N \leq 10.000$, $1 <= M <= 10.000$), representando respectivamente o número de faixas de premiações e o número de ogros cuja força foi medida.

A segunda linha de um caso de teste contem $N-1$ inteiros A_i ($A_{i-1} < A_i < A_{i+1}$, $1 \leq A_i \leq 1.000.000.000$). A primeira faixa de premiação é dada por forças menores que A_1. A i-ésima faixa de premiação é composta das forças que são maiores ou iguais a A_{i-1} e menores que A_i. A n-ésima pontuação é composta das forças maiores ou iguais a A_{N-1}.

A terceira linha contem N inteiros F_i ($1 \leq F_i \leq 1.000.000.000$, $F_{i-1} < F_i < F_{i+1}$) em ordem crescente, representando a premiação de cada faixa de premiação, nesta ordem.

A quarta e última linha de um caso de teste contêm M inteiros O_i ($1 \leq O_i \leq 1.000.000.000$), um para cada ogro, representando qual força cada ogro alcançou.

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha, contendo M inteiros, um para cada ogro, na ordem dada pela entrada, representando qual premiação cada ogro recebeu pela sua força alcançada.

Exemplo de entrada	**Exemplo de saída**

`3 4`
`3 5`
`1 4 7`
`2 3 9 4`
	`1 4 7 4`

Exemplo de entrada	**Exemplo de saída**

`2 10`
`4`
`5 200`
`1 3 4 5 5 6 2 1 3 4`
	`5 5 200 200 200 200 5 5 5 200`

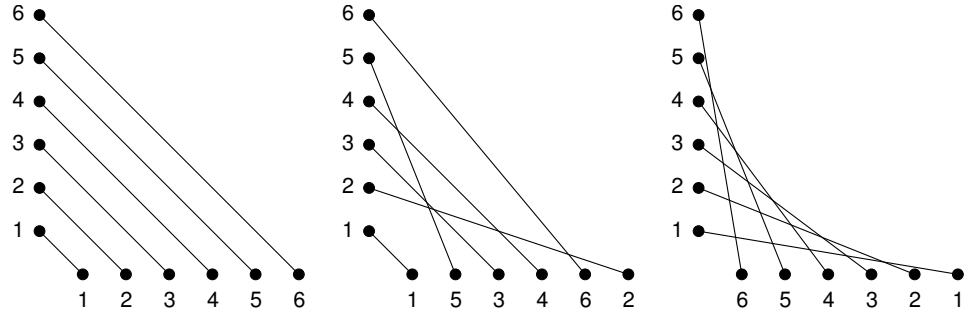
Exemplo de entrada	**Exemplo de saída**

`10 1`
`1 2 3 4 5 6 7 8 9`
`1 10 100 101 102 103 104 105 106 200`
`5`
	`103`

Linhas Cruzadas

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

Uma das atividades de recreação preferidas de Letícia é compor desenhos com linhas coloridas esticadas entre preguinhos numa base de madeira. Quanto mais cruzamentos entre pares de linhas, mais interessante fica a figura. Neste problema temos N pregos na vertical e N pregos na horizontal, como na figura abaixo. Os pregos na vertical possuem uma numeração fixa, de 1 a N, de baixo para cima. Os pregos na horizontal também são numerados de 1 a N, mas a ordem pode ser qualquer uma. Letícia vai sempre esticar uma linha entre cada par de pregos que tiverem o mesmo número. Dada a ordem dos pregos horizontais, seu programa deve computar o número total de cruzamentos entre pares de linhas no desenho de Letícia. Por exemplo, os três desenhos da figura possuem, respectivamente, 0, 6 e 15 cruzamentos.



Entrada

A primeira linha da entrada contém um número natural N. A segunda linha contém N números naturais distintos de 1 a N, representando a ordem dos pregos na horizontal.

Saída

Seu programa deve escrever uma linha na saída, contendo o número de cruzamentos entre pares de linhas, conforme a descrição anterior.

Restrições

- $2 \leq N \leq 60000$

Informações sobre a pontuação

- Em um conjunto de casos de teste somando 60 pontos, $N \leq 30000$

Exemplos

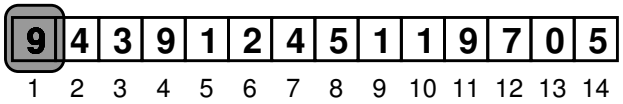
Entrada	**Saída**
6 1 5 3 4 6 2	6

Entrada	**Saída**
15 5 8 15 12 2 1 9 7 4 11 14 10 3 6 13	49

Segredo do Cofre

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

O sistema de segredo para abrir esse cofre é bastante complexo. Ao invés de girar um botão várias vezes, como a gente vê normalmente nos filmes, o dono do cofre tem que deslizar um controle para a esquerda e para a direita, em cima de uma barra, várias vezes, parando em determinadas posições. A barra possui N posições e cada posição contém um número inteiro entre 0 e 9, inclusive. No exemplo da figura, a barra tem 14 posições e o controle está na posição 1.



O segredo vai depender de quantas vezes cada um dos dez inteiros entre 0 e 9 vai aparecer dentro do controle. Por exemplo, suponha que o dono deslize o controle da posição inicial 1 até a posição 9, depois para a posição 4, depois para a posição 11 e por fim até a posição 13. Veja que o inteiro 1, por exemplo, vai aparecer seis vezes dentro do controle; e o inteiro 9 vai aparecer quatro vezes.

Dada a sequência de inteiros na barra e a sequência de posições entre as quais o dono desliza o controle, começando da posição inicial 1, seu programa deve contar quantas vezes cada inteiro, entre 0 e 9, vai aparecer dentro do controle.

Entrada

A primeira linha da entrada contém dois inteiros N e M, representando o número de posições na barra do cofre e o número de posições na sequência que o dono vai seguir para deslizar o controle. A segunda linha contém N inteiros entre 0 e 9, definindo a barra do cofre. A terceira linha contém M inteiros representando a sequência de posições que o dono vai seguir. A primeira posição nessa sequência é sempre 1 e não há duas posições consecutivas iguais.

Saída

Seu programa deve imprimir uma linha contendo 10 inteiros, representando o número de vezes que cada inteiro, entre 0 e 9, vai aparecer no controle da barra.

Restrições

- $2 \leq N \leq 10^5$ e $2 \leq M \leq 10^5$

Informações sobre a pontuação

- Em um conjunto de testes somando 40 pontos, $N \leq 1000$ e $M \leq 1000$

Exemplos

Entrada	**Saída**
14 5 9 4 3 9 1 2 4 5 1 1 9 7 0 5 1 9 4 11 13	1 6 3 1 4 3 0 1 0 4

Entrada	**Saída**
5 4 5 8 0 5 1 1 4 2 5	3 1 0 0 0 3 0 0 2 0