Pão a Metro

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

Pão a metro é um tipo de sanduíche gigante que é uma excelente opção de lanche para torneios de programação, embora a experiência já tenha mostrado que o oferecimento de sanduiches pode gerar reclamação dos competidores. Outro grande problema é que algumas pessoas são mais gulosas que outras e, dessa maneira, acabam pegando pedaços maiores que os pedaços dos outros. Para a final da OBI, a coordenação estava pensando em providenciar pão a metro para os competidores, porém tais problemas os fizeram recuar na idéia.

Embora a idéia tenha sido momentaneamente abandonada, uma idéia simples surgiu: cortar previamente o pão em fatias de tamanho iguais e distribuí-las entre as pessoas. O único problema com tal idéia é que se o número de pessoas for muito grande, fica impraticável ter apenas um pão. Por exemplo, se quisermos que 1.000 pessoas recebam 20 centímetros de sanduíche, seria necessário um sanduíche de 20.000 centímetros, ou 200 metros!

Alguém levantou a seguinte hipótese: se houvessem N pessoas e fossem encomendados K sanduíches de empresas diferentes, cada qual com uma determinada metragem (tamanho) M_i ($1 \leq i \leq K$), seria possível retirar desses pães N fatias de mesmo tamanho, possivelmente sobrando partes nao utilizadas. A questão seria: qual o tamanho inteiro máximo que essas fatias poderão ter?

Por exemplo, se tivermos $K = 4$, com os tamanhos (em centímetros) $M_1 = 120$, $M_2 = 89$, $M_3 = 230$ e $M_4 = 177$ e $N = 10$, podemos retirar N fatias iguais de tamanho máximo 57, pois assim conseguimos 2 fatias no primeiro pão, 1 no segundo, 4 no terceiro e 3 no quarto, totalizando as 10 fatias necessárias. Se tentarmos cortar fatias de tamanho 58, só será possível obter 3 fatias do terceiro pão, totalizando 9 e, portanto, 57 é realmente o melhor que podemos obter. Note que não podemos usar duas ou mais fatias menores de diferentes pães para formarmos uma fatia do tamanho selecionado. (ficaria muito deselegante dar um lanche recortado às pessoas).

Tarefa

Escreva um programa que, dados os tamanhos de pão disponíveis (em centímetros) e a quantidade de pessoas a serem atendidas, retorne o tamanho inteiro máximo (em centímetros) da fatia que pode ser cortada de maneira a atender todas as pessoas.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado). A primeira linha da entrada contém um inteiro N que indica a quantidade pessoas ($1 \leq N \leq 10.000$). A segunda linha contém um inteiro K ($1 \leq K \leq 10.000$) que é a quantidade de sanduíches disponível. Na terceira linha há K inteiros M ($1 \leq M \leq 10.000$) separados por um espaço em branco representando o tamanho de cada pão.

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha, contendo o tamanho inteiro máximo da fatia que pode ser cortada.

Entrada	**Entrada**	**Entrada**

`10`
`4`
`120 89 230 177`
	`3`
	`2`
	`45 85`
		`7`
		`7`
		`100 98 99 505 102 97 101`

Saída	**Saída**	**Saída**
`57`	`42`	`101`

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`

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