Operação `aumentar_chave`

Escreva uma função `aumentar_chave(heap, pos, novo)` que faz a operação `aumentar_chave` em uma *heap* **mínima**.

O parâmetro `heap` deve referenciar uma lista que é uma *heap* mínima. O parâmetro `pos` deve ser a posição na qual queremos "aumentar" o valor da chave. Já o parâmetro `novo` deve conter o novo valor da chave nessa posição.

Como se trata de uma *heap* mínima, então na verdade o novo valor deve ser igual ou **menor** do que o valor anterior.

Se o parâmetro `pos` for igual ao tamanho da lista, então uma nova chave deve ser inserida no fim da *heap*.

A operação `aumentar_chave` deve trocar o elemento `heap[pos]` com o seu pai, repetidamente, até que o elemento em `heap[pos]` não esteja na primeira posição da *heap* ou não esteja mais violando a propriedade de *heap* mínima com seu pai.

Entrada e Saída

A entrada conterá vários casos de teste.

Cada caso de é descrito em três linhas. A primeira contém os elementos de uma *heap* mínima. A segunda contém a posição na qual queremos aplicar a operação `aumentar_chave`. A terceira contém o novo valor da chave que deve ser inserida nessa posição.

A entrada termina quando aparecer uma lista vazia.

Para cada caso de teste, imprima uma lista indicando a *heap* resultante após o "aumento" da chave.

Exemplos de Entrada e Saída

Entrada	`[2, 6, 7]`
	`1`
	`3`
	`[3, 5, 4]`
	`1`
	`2`
	`[2, 7, 6]`
	`3`
	`4`
	`[]`

Saída	[2, 3, 7] [2, 3, 4] [2, 4, 6, 7]
Entrada	[2, 4, 3, 8, 5, 6, 7, 8] 3 1 [2, 3, 4, 6, 4, 5, 8, 7, 7, 4] 5 1 [2, 3, 3, 4, 8, 5, 9, 7, 9, 9] 10 1 []
Saída	[1, 2, 3, 4, 5, 6, 7, 8] [1, 3, 2, 6, 4, 4, 8, 7, 7, 4] [1, 2, 3, 4, 3, 5, 9, 7, 9, 9, 8]