Allineare due liste di interi per massimizzare la somma dei minimi (`lists_alignment`)

Nella nostra agenzia matrimoniale, lavoriamo con liste di numeri interi non-negativi, da cui ci risulta possibile rimuovere elementi in posizione arbitraria pur di non alterare l'ordine tra gli elementi rimanenti. Ad esempio, rimuovendo gli elementi di posizione dispari dalla lista $L = 10, 20, 30, 40, 50, 60$, otteniamo la lista $L' = 20, 40, 60$, dove l'ordine degli elementi mantenuti in lista è rimasto lo stesso. La lunghezza $\ell(L)$ di una lista $L = L_1, L_2, ..., L_{\ell(L)}$ è il numero dei suoi elementi. Quando una lista U di uomini ed una lista D di donne hanno la stessa lunghezza $\ell = \ell(U) = \ell(D)$ il *pay-off* è dato da $\text{val}(U, D) := \sum_{i=1}^{\ell} \min(U_i, D_i)$. Date in input le due liste U e D, estrai da esse le sottoliste U' e D' per rimozione di zero o più elementi, con l'obiettivo di massimizzare $\text{val}(U', D')$.

Input

Si legga l'input da `stdin`. La prima riga contiene T, il numero di testcase (istanze) da risolvere. Seguono T istanze del problema. Ciascuna istanza è descritta in tre righe, dove la prima riga contiene gli interi $\ell(U)$ e $\ell(D)$ separati da spazi, la seconda riga contiene la lista U (ovvero gli elementi della lista U nel loro ordine e separati da spazi), la terza ed ultima riga contiene la lista D.

Output

Per ciascuna istanza, prima di leggere l'istanza successiva, scrivi su `stdout` il tuo output. Esso consta di tre righe che codificano la tua soluzione (U', D'): la prima riga contiene l'intero $\text{val}(U', D')$, la seconda riga contiene la lista U', la terza ed ultima riga contiene la lista D'. Per le istanze dei subtask facilitati (taggati `only_val`) basta che la prima di queste tre righe riporti il valore ottimo corretto affinchè venga assegnato il punto (invece negli altri subtask verificheremo anche che nelle restanti due righe sia offerta una soluzione valida del valore di pay-off ottimo dichiarato nella prima riga).

Esempio

Input da `stdin`

```
2
 5 5
 9 1 1 1 1
 1 9 1 1 1
 7 8
 101 102 103 104 105 106 107
  21  52  33  74  15  46  67  88
```

Output su `stdout`

```
12
9 1 1 1
9 1 1 1
381
101 102 103 104 105 106 107
 21  52  33  74  46  67  88
```

Subtask

Il tempo limite per istanza (ossia per ciascun testcase) è sempre di 1 secondo.

I testcase sono raggruppati nei seguenti subtask.

1. **[2 istanze] `esempi_testo`:** i due esempi del testo
2. **[8 istanze] `tiny_only_val`:** $\ell(U), \ell(V) \leq 10$
3. **[4 istanze] `tiny_with_sol`:** $\ell(U), \ell(V) \leq 10$
4. **[12 istanze] `small_only_val`:** $\ell(U), \ell(V) \leq 20$
5. **[6 istanze] `small_with_sol`:** $\ell(U), \ell(V) \leq 20$
6. **[12 istanze] `medium_only_val`:** $\ell(U), \ell(V) \leq 100$
7. **[6 istanze] `medium_with_sol`:** $\ell(U), \ell(V) \leq 100$
8. **[16 istanze] `big_easy_only_val`:** $\ell(U), \ell(V) \leq 500, \max_{u \in U} \leq \min_{d \in D}$
9. **[8 istanze] `big_easy_with_sol`:** $\ell(U), \ell(V) \leq 500, \max_{u \in U} \leq \min_{d \in D}$

10. **[4 istanze] `big_only_val`:** $\ell(U), \ell(V) \leq 500$
11. **[2 istanze] `big_with_sol`:** $\ell(U), \ell(V) \leq 500$

In generale, quando si richiede la valutazione di un subtask vengono valutati anche i subtask che li precedono, ma si evita di avventurarsi in subtask successivi fuori dalla portata del tuo programma che potrebbe andare in crash o comportare tempi lunghi per ottenere la valutazione completa della sottomissione. Ad esempio, chiamando[1, 2] :

```
rtal -s <URL>  connect -x <token> -a size=small_with_sol
       lists_alignment  -- python my_solution.py
```

vengono valutati, nell'ordine, i subtask:

`esempi_testo`, `tiny_only_val`, `tiny_with_sol`, `small_only_val`, `small_with_sol`.

Il valore di default per l'argomento `size` è `big_with_sol` che include tutti i testcase.

[1] `<URL>` server esame: `wss://ta.di.univr.it/esame`

[2] `<URL>` server esercitazioni e simula-prove: `wss://ta.di.univr.it/algo`