Composizione di una somma dovuta (`conio3` [62 punti])

Se i tagli di monete e banconote disponibili sono quelli standard (1€, 2€, 5€, 10€, 20€, 50€, 100€, 200€, 500€, 1000€), e disponiamo di almeno due pezzi di ciascun taglio, possiamo allora comporre qualsiasi somma che non ecceda il valore totale in cassa. Con disponibilità illimitata di pezzi di ogni taglio possiamo comporre qualsiasi somma ed un naturale algoritmo greedy lo fa impiegando sempre meno pezzi possibile. Nella realtà non disponiamo però di una quantità illimitata di pezzi. Affronteremo quì la situazione più generale dove il set di tagli disponibili, con la molteplicità di ciascuno di essi, siano dati in input assieme al valore della somma da saldare.

Input

Si legga l'input da `stdin`. La prima riga contiene T, il numero di testcase (istanze) da risolvere. Ogni istanza ha il seguente formato:

S (int)	D (int)		
T_1 (int)	T_2 (int)	...	T_D (int)
N_1 (int)	N_2 (int)	...	N_D (int)

dove, nella prima riga, il token S indica la somma da saldare (un numero naturale) e D il numero di diversi tagli resi disponibili dalla zecca di quel paese (un secondo numero naturale). La seconda riga specifica i valori dei tagli della zecca, questi sono D numeri interi positivi. Per $i \in [1, D]$, l'i-esimo token N_i nella terza riga è il numero di pezzi di taglio T_i disponibili in cassa (un numero naturale, potrebbe anche essere zero).

Output

Per ciascuna istanza, prima di leggere l'istanza successiva, scrivi su `stdout` il tuo output così strutturato:

X_1 (int)	X_2 (int)	...	X_D (int)

ossia una singola riga di D token che specificano quanti pezzi X_i impiegare di ciascun taglio $i = 1, ..., D$. Pur volendolo minimizzare, ci si impegna a che l'esborso totale $E = \sum_{i=1}^{D} T_i X_i$ copra la somma dovuta ($E \geq S$). Tra le soluzioni ad esborso minimo, se ne produca una che minimizzi inoltre il numero di pezzi che passano di mano ($\sum_{i=1}^{D} X_i$). Ove quanto in cassa non basti a coprire la somma S dovuta si consegni tutto ciò che si ha.

Esempio di Input/Output

Input da `stdin`

```
6
98 6
  1   2   5  10  20  50
100 100 100 100 100 100
98 6
  1   2   5  10  20  50
100   0 100   1 100   0
98 6
  1   2   5  10  20  50
100   0   0   0 100   0
98 6
  1   2   5  10  20  50
```

Output su `stdout`

```
98 6
1 1 1 0 2 1
98 9
3 0 1 1 4 0
98 22
18 0 0 0 4 0
100 2
0 0 0 0 0 2
52357 478
30 43 73 31 51 19 22 209
20488 302
30 44 73 31 41 11 12 60
```

```
  0   1 100 100 100 100
52357 8
 5 12 17 33 71 113 151 193
30 44 73 31 51  19  22 300
52357 8
 5 12 17 33 71 113 151 193
30 44 73 31 41  11  12  60
```

Subtask

Per ogni istanza viene valutato il raggiungimento di due goal:

`mincost` questo primo goal viene raggiunto se si minimizza il valore totale dell'esborso $E \geq S$

`mincoins_at_mincost` questo secondo goal viene raggiunto se, raggiunto il primo goal, si minimizza inoltre il numero di pezzi di denaro coinvolti nel comporre l'esborso E

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

I testcase sono raggruppati nei seguenti subtask.

1. **[12 pts← 6 istanze da 1 + 1 punti] `esempi_testo`:** i sei esempi del testo
2. **[10 pts← 5 istanze da 1 + 1 punti] `tiny`:** $S \leq 1€$
3. **[10 pts← 5 istanze da 1 + 1 punti] `small`:** $S \leq 10€$
4. **[8 pts← 4 istanze da 1 + 1 punti] `medium`:** $S \leq 100€$
5. **[6 pts← 3 istanze da 1 + 1 punti] `big`:** $10.000€$
6. **[8 pts← 2 istanze da 2 + 2 punti] `large`:** $1.000.000€$
7. **[8 pts← 2 istanze da 2 + 2 punti] `extra_large`:** $S \leq 1.000.000.000€$

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
       conio3  -- python my_solution.py
```

vengono valutati, nell'ordine, i subtask:

`esempi_testo`, `tiny`, `small`.

Il valore di default per l'argomento `size` è `extra_large` 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`