coprire una somma minimizzando lo spreco (`conio2` [60 punti])

Abbiamo due soli tipi di monete, quelle d'oro, di valore V_1 e quelle di platino, di valore V_2. Dobbiamo versare una certa somma S, possiamo dare di più ma non di meno, cerchiamo di sperperare il meno possibile.

Input

Per descrivere una singola istanza, i tre valori S, V_1 e V_2 (interi positivi tutti e tre) sono scritti, in questo ordine e separati da spazio, entro un'unica riga. L'input è letto da `stdin`, la cui prima riga conterrà T, il numero di testcase (istanze) da risolvere. Ogni istanza prende una diversa riga e così il server scriverà un totale di $T + 1$ righe. Dalla seconda in poi, prima di emettere la riga successiva, attenderà la tua risposta per quell'istanza.

Output

Per ciascuna istanza, prima di leggere l'istanza successiva o per concludere la sessione col server, scrivi su `stdout` un'unica riga contentente i tre numeri naturali V, n_1 e n_2 separati da spazi. Quì V indica la minima somma che dovrai versare, mentre n_1 ed n_2 sono tali che $V = V_1 n_1 + V_2 n_2$ e precisano quante monete di ciascun tipo vadano consegnate. Se la riga rispetta questo formato (tre numeri), ti potranno essere riconosciuti alcuni o tutti dei seguenti tre *goals*:

/`optval`: se V è davvero la minima somma che ti tocca di versare.

/`optsol`: se $V_1 n_1 + V_2 n_2$ è la minima somma che ti tocca di versare per soddisfare $V_1 n_1 + V_2 n_2 \geq S$.

/`bestoptsol`: se $V_1 n_1 + V_2 n_2$ è la minima somma che ti tocca di versare e il numero di monete impiegato, $n_1 + n_2$, è il più piccolo possibile.

Esempio di Input/Output

Input da `stdin`

```
4
98
324
3472
3456789
```

Output su `stdout`

```
6
1 1 1 0 2 1 0 0 0 0
5
0 2 0 0 1 0 1 1 0 0
8
0 1 0 0 1 1 0 2 0 3
1464
0 2 1 1 1 1 0 1 1 1456
```

Subtask

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

I testcase sono raggruppati nei seguenti subtask.

1. **[12 pts← 4 istanze da 1 + 1 + 1 punti] `esempi_testo`:** i quattro esempi del testo
2. **[15 pts← 5 istanze da 1 + 1 + 1 punti] `small`:** $V_1, V_2, S \leq 30€$
3. **[12 pts← 4 istanze da 1 + 1 + 1 punti] `medium`:** $V_1, V_2, S \leq 300€$
4. **[9 pts← 3 istanze da 1 + 1 + 1 punti] `big`:** $V_1, V_2, S \leq 10.000€$
5. **[6 pts← 2 istanze da 1 + 1 + 1 punti] `large`:** $V_1, V_2, S \leq 1.000.000€$
6. **[6 pts← 2 istanze da 1 + 1 + 1 punti] `extra_large`:** $V_1, V_2, S \leq 1.000.000.000€$

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=medium
        conio2  -- python my_solution.py
```

vengono valutati, nell'ordine, i subtask:

`esempi_testo`, `small`, `medium`.

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`