Count, rank e unrank di ombre crescenti (`quanti_poldo` [100 punti])

Per ogni numero naturale $n \in \mathbb{N}$ sia $\mathbb{N}_n := \{n' \in \mathbb{N} : n' < n\}$. Una sequenza $S = s_0, s_1, ..., s_{n-1}$ è detta *(monotona) crescente* se $s_{i+1} > s_i$ per ogni $i \in \mathbb{N}_{n-1}$. Per ogni $k \in \mathbb{N}_n$, ogni sequenza crescente $I : \mathbb{N}_k \longrightarrow \mathbb{N}_n$ definisce una diversa sottosequenza di S di lunghezza k; più precisamente, la sequenza di k indici $I = i_0, i_1, ..., i_{k-1}$ con $0 \leq i_0 < i_1 < ... < i_{k-1} < n$ definisce la sequenza $S \circ I := s_{i_0}, s_{i_1}, ..., s_{i_{k-1}}$ come una sottosequenza di S di lunghezza k. Se $S \circ I$ è una sottosequenza crescente di S allora I è detta un *ombra crescente per* S, o, più brevemente, un *ombra di* S. Indichiamo con $\mathbb{I}_S$ l'insieme di tutte le ombre crescenti per S. Su $\mathbb{I}_S$ vige un'ordinamento totale che esemplifichiamo per $n = 3$ assumendo una S crescente qualsiasi:

```
sequenza    rango
0 1 2       0
0 1         1
0 2         2
0           3
1 2         4
1           5
2           6
            7
```

Se ti restano dubbi su come le ombre crescenti per S siano ordinate per altre sequenze S puoi interrogare il servizio `list` offerto per questo problema `TALight` fornendo una tua sequenza S in input.

Ti chiediamo di produrre del codice che implementi le seguenti competenze:

counting data S, calcolare il numero $f(S)$ di ombre di S.

ranking date S e $I \in \mathbb{I}_S$, restituire il rango r di I in $\mathbb{I}_S$, ossia l'intero nell'intervallo $[0, f(S) - 1]$ che specifica la posizione di I nel nostro ordinamento (si parte da 0).

unranking dati S e r, con $r \in [0, f(S) - 1]$, restituire la sequenza I in $\mathbb{I}_S$ che, partendo da 0, appare in posizione r entro il nostro ordinamento su $\mathbb{I}_S$.

Nota: `rank(unrank(S, r)) = r`.

Assunzioni

1. quando ti chiediamo di computare $f(S)$, in realtà, se vorrai, potrai limitarti a restituire il resto della divisione di $f(S)$ per $1.000.000.007$. Con questo obiettivo puoi acquisire maggiore efficienza che ti è necessaria per risolvere le istanze più grandi.

2. quando ti chiediamo di fare `rank`, è nostra cura fornirti in input una $I \in \mathbb{I}_S$ di rango inferiore a $1.000.000.007$.

3. quando ti chiediamo di fare `unrank`, è nostra cura fornirti in input un $r < 1.000.000.007$.

NB: per rank e unrank, anche se il rango coinvolto è inferiore $1.000.000.007$, $f(S)$ potrebbe essere maggiore, quindi, se utilizzi i numeri modulo $1.000.000.007$ dovresti ricordare quantomeno se il vero valore di $f(S)$ è maggiore di $1.000.000.007$, per effettuare correttamente tutti i confronti necessari.

Input

Si legga l'input da `stdin`. La prima riga contiene T, il numero di testcase (istanze) da risolvere. Seguono T istanze del problema, dove ogni istanza può porre diverse domande per uno stesso valore di n. Per ogni istanza, la prima riga contiene quattro numeri interi separati da uno spazio: n, c, r ed u, dove $c \in \{0, 1\}$ indica se si richiede il valore di $f(S)$, r è il numero di richieste di ranking e u è il numero di richieste di unranking. La seconda riga contiene gli n numeri interi costituenti S, nell'ordine $s_0, s_1, ..., s_{n-1}$ e separati da spazio. Seguono r righe, l'i-esima delle quali contiene una sequenza I_i di $\mathbb{I}_S$. Infine una riga che contiene u numeri interi $r_1, r_2, ..., r_u \in [0, f(S) - 1]$ separati da spazio. (Questa riga sarà vuota, ma comunque presente, del caso u sia 0.)

Output

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

1. la prima riga contiene il numero $f(S)$. Per le istanze con $c = 0$, essa viene di fatto ignorata dal correttore, ma devi comunque stampare una riga (puoi ad esempio stampare la riga vuota, oppure stampare comunque il numero corretto di formule). Quando invece $c = 1$ teniamo buona come risposta sia $f(S)$ che $f(S)\%1.000.000.007$ ma col primo potresti non riuscire a risolvere le istanze più grandi.
2. la seconda riga contiene r numeri nell'intervallo $[0, f(S) - 1]$ separati da spazio, l'i-esimo di questi numeri vuole essere il rango di I_i, ossia dell'i-esima sequenza di $\mathbb{I}_S$ ricevuta in input per questa istanza. (Di nuovo, il contenuto di questa riga verrà ignorato per ogni istanza con $r = 0$.)
3. seguono $u \geq 0$ righe l'i-esima delle quali contiene l'ombra di S di rango r_i, dove r_i è l'i-esimo numero contenuto nell'ultima riga ricevuta in input per questa istanza.

Esempio

Input

```
3
6 1 0 0
1 6 2 5 3 4

6 1 6 0
1 6 2 5 3 4
0 1
0 2 3
0 2 4
0 4 5
0 5


6 1 0 6
1 6 2 5 3 4
0 1 3 21 7 9
```

Output

```
22

22
0 1 3 7 9 21

```

```
22

0 1
0 2 3
0 2 4

0 4 5
0 5
```

Subtask

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

I testcase sono raggruppati nei seguenti subtask.

1. **[3 pts← 3 istanze da 1 punto] `esempi_testo`:** i tre esempi del testo
2. **[9 pts← 9 istanze da 1 punto] `small_c`:** $n \leq 10, c = 1, r = 0, u = 0$
3. **[11 pts←11 istanze da 1 punto] `small_r`:** $n \leq 10, c = 0, u = 0, r \leq 20$
4. **[11 pts←11 istanze da 1 punto] `small_u`:** $n \leq 10, c = 0, r = 0, u \leq 20$
5. **[11 pts←11 istanze da 1 punto] `medium_c`:** $n \leq 20, c = 1, r = 0, u = 0$
6. **[11 pts←11 istanze da 1 punto] `medium_r`:** $n \leq 20, c = 0, u = 0, r \leq 50$
7. **[11 pts←11 istanze da 1 punto] `medium_u`:** $n \leq 20, c = 0, r = 0, u \leq 50$
8. **[11 pts←11 istanze da 1 punto] `big_c`:** $n \leq 200, c = 1, r = 0, u = 0$
9. **[11 pts←11 istanze da 1 punto] `big_r`:** $n \leq 200, c = 0, u = 0, r \leq 200$
10. **[11 pts←11 istanze da 1 punto] `big_u`:** $n \leq 200, c = 0, r = 0, u \leq 500$

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 -a size=medium_c quanti_poldo -- <MY_SOLUTION>
```

vengono valutati, nell'ordine, i subtask:

`esempi_testo`, `small_c`, `small_r`, `small_u`, `medium_c`.

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

Nel template di riga di comando quì sopra `<MY_SOLUTION>` è una qualsiasi scrittura che ove immessa anche da sola al prompt della CLI comporti l'avvio del solver da tè realizzato. Solo alcuni esempi:

- `./a.out` per un compilato da `C/c++`, eventualmente seguito dagli argomenti che prevede
- `./my_solution.py arg1 arg2 ...` se il tuo file `my_solution.py` col codice python ha i permessi di esecuzione e inizia con la riga di shebang
- `python my_solution.py` o `python3 my_solution.py` per far eseguire il tuo script da un interprete python

Se vuoi che una tua sottomissione venga conteggiata ai fini di un esame o homework devi allegare il sorgente della tua soluzione con `-fsource=<FILENAME>` (ad esempio `-fsource=my_solution.py` subito a valle del comando `connect`. Inoltre devi aver precedentemente affettuato il login tramite credenziali GIA (lancia prima `rtal -s <URL> login` e segui le istruzioni per impostare il sign-on).

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

[2] `<URL>` server esercitazioni e simula-prove: `wss://ta.di.univr.it/algo2025`

Il comando `rtal` prevede diversi parametri, consigliamo di esplorarne e sperimentarne le potenzialità ed opzioni d'uso. Un tutorial all'uso di `rtal` è esposto alla pagina:

`https://github.com/romeorizzi/AlgoritmiUniVR/tree/main/strumenti`

Servizi TALight a tua disposizione

Se reputi ti serva disambiguare meglio l'ordine da noi stabilito su $\mathbb{I}_S$, puoi chiedere a TALight, ad esempio:

```
rtal -s <URL>  connect -x <token> quanti_poldo list  -a S="4 2 15 3 7"
```

Ulteriori servizi di supporto alla tua esplorazione del problema (esempi di chiamate):

```
rtal -s <URL>  connect -x <token> quanti_poldo check_count  -a S="4 2 15 3 7" -a risp=5
```

```
rtal -s <URL>  connect -x <token> quanti_poldo check_rank  -a S="4 2 15 3 7" -a input_shadow="1 3" -a right_rank=8
```

```
rtal -s <URL>  connect -x <token> quanti_poldo check_unrank -a input_rank=3  -a S="4 2 15 3 7" -a input_rank=2 -a right_shadow="2 4 5"
```

Nota: questi ulteriori servizi ti risponderanno con un certo ritardo per non avere un effetto controproducente sul piano della tua esplorazione autonoma del problema.