Count, rank e unrank delle formule di parentesi ben formate trasparenti (`FBF_trasparenti`)

Ricordiamo che una formula di parentesi ben formata (FBF) è una stringa sull'alfabeto $\Sigma = \{\,\mathbf{(}, \mathbf{)}\,\}$ che può essere generata partendo dal simbolo non-terminale **S** tramite applicazione non-deterministica delle seguenti regole di sostituzione:

1. $\mathbf{S} \longrightarrow \mathbf{(S)}$
2. $\mathbf{S} \longrightarrow \mathbf{SS}$
3. $\mathbf{S} \longrightarrow \varepsilon$

dove ε indica la stringa vuota, e la terza regola è quindi necessaria per sbarazzarsi del simbolo non-terminale **S**. Ad esempio, le formule **()()** e **(())** vengono entrambe ottenute dalla stringa iniziale **S** con una singola applicazione della regola 2, seguita da due applicazioni sia della 1 che della 3.

Una FBF è detta *trasparente* se ammette una derivazione in cui la regola 2 non viene mai applicata ad un simbolo **S** che è stato introdotto a sua volta con la regola 2 ($\longrightarrow$ essere trasparenti è una **NP**-property).

Equivalentemente, una FBF è detta *trasparente* se **non** contiene una sottostringa che ricada nel template $\mathbf{(}A\mathbf{)(}B\mathbf{)(}C\mathbf{)}$ dove A, B e C sono FBF ($\longrightarrow$ essere trasparenti è una **coNP**-property).

La più piccola FBF non trasparente è appunto:

()()()

ed ogni FBF non trasparente in un certo qual senso la contiene.

Prese insieme, queste due definizioni equivalenti vorrebbero caratterizzare il concetto, ossia offrirti un'idea abbastanza chiara di cosa sia una FBF trasparente. Ma in realtà, ai nostri scopi, dovrebbe in tutto bastarti una comprensione anche solo intuitiva, che puoi ad esempio raccogliere guardando al seguente listing delle FBF trasparenti per $n = 4$. Poichè siamo interessati nel gestire le FBF trasparenti adottando un preciso criterio di ordinamento lessicografico sull'applicazione delle regole di generazione, cerca di estrapolare dall'esempio il criterio di ordinamento quì adottato (l'esempio dovrebbe bastarti, e anzi ti conviene assicurarti di spremerlo, ma in caso contrario puoi sempre avvalerti dei servizi TALight messi a disposizione col problema):

```
(((()))) 
((()())) 
((())()) 
(()(())) 
((()))() 
(()())() 
(())(()) 
()((())) 
()(()())
```

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

counting dato n, calcolare il numero $f(n)$ delle FBF trasparenti con n coppie di parentesi.

ranking data una FBF trasparente t, restituire il suo rango r, ossia il numero intero nell'intervallo $[0, f(n) - 1]$ che specifica la posizione di t entro il nostro ordinamento (si parte da 0).

unranking dati n ed r, con $r \in [0, f(n) - 1]$, restituire la FBF trasparente che, partendo da 0, appare in posizione r entro il nostro ordinamento delle FBF trasparenti con n coppie di parentesi.

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

Assunzioni

1. quando ti chiediamo di computare $f(n)$, in realtà, se vorrai, potrai limitarti a restituire il resto della divisione di $f(n)$ 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 FBF 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 $1.000.000.007$, $f(n)$ potrebbe essere maggiore, quindi, se utilizzi i numeri modulo $1.000.000.007$ dovresti ricordare quantomeno se il vero valore di $f(n)$ 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 $c = 0, 1$ indica se si richiede il valore di $f(n)$, r è il numero di richieste di ranking e u è il numero di richieste di unranking. Seguono r righe, l'i-esima delle quali contiene una FBF trasparente t_i che ha n coppie di parentesi. Infine una riga che contiene u numeri interi $r_1, r_2, ..., r_u \in [0, f(n) - 1]$ separati da spazio. (Questa riga sarà vuota 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 $f(n)$. 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 $f(n)$ che $f(n)\%\ 1.000.000.007$ ma col primo potresti non riuscire a risolvere le istanze più grandi.
2. la seconda riga contiene r numeri nell'intervallo $[0, f(n) - 1]$ separati da spazio, l'i-esimo di questi numeri vuole essere il rango di t_i, ossia dell'i-esima FBF trasparente ricevuta in input per questa istanza. (Di nuovo, il contenuto di questa riga verrà ignorato per le istanza con $r = 0$.)
3. seguono $u \geq 0$ righe l'i-esima delle quali contiene l'FBF trasparente di n coppie di parentesi 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
4 1 0 0

4 0 3 0
((((()))
((()(()))
()(()())


```

```
4 0 0 3
0 3 7
```

Output

```
9

9
0 3 8
9

(((()))))
((()(())))
()((()))
```

Subtask

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

I testcase sono raggruppati nei seguenti subtask.

1. **[3 istanze] `esempi_testo`:** i due esempi del testo
2. **[9 istanze] `small_c`:** $n \leq 10, c = 1, r = 0, u = 0$
3. **[11 istanze] `small_r`:** $n \leq 10, c = 0, u = 0, r \leq 20$
4. **[11 istanze] `small_u`:** $n \leq 10, c = 0, r = 0, u \leq 20$
5. **[11 istanze] `medium_c`:** $n \leq 20, c = 1, r = 0, u = 0$
6. **[11 istanze] `medium_r`:** $n \leq 20, c = 0, u = 0, r \leq 50$
7. **[11 istanze] `medium_u`:** $n \leq 20, c = 0, r = 0, u \leq 50$
8. **[11 istanze] `big_c`:** $n \leq 200, c = 1, r = 0, u = 0$
9. **[11 istanze] `big_r`:** $n \leq 200, c = 0, u = 0, r \leq 200$
10. **[11 istanze] `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 -x <token> -a size=medium_c
    FBF_trasparenti  -- python my_solution.py
```

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.

Servizi TALight a tua disposizione

Se reputi ti serva chiarire la nozione di trasparenza, puoi chiedere a TALight se una certa FBF sia trasparente, ad esempio:

```
rtal -s <URL>  connect FBF_trasparenti is_transparentFBF -a FBF="(()(()())())"
```

Il servizio non solo ti risponderà con un valore di verità, ma cercherà anche di certificarlo ai tuoi occhi.

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

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

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

```
    rtal -s <URL>  connect FBF_trasparenti check_num_transparentFBFs -a n_pairs=4 -a
risp=9

    rtal -s <URL>  connect FBF_trasparenti check_rank -a input_FBF="(()(()))"  -a
right_rank=3

   rtal -s <URL>  connect FBF_trasparenti check_unrank -a input_rank=3  -a right_FBF="(()
(()))"
```

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