Triangolo (Discesa): solo computo del valore ottimo di una soluzione

Abbiamo inserito una moneta in un flipper a premi. I premi, ciascuno con un proprio valore, sono disposti in un triangolo verticale, ciascuno appoggiato su un suo piolo di supporto.

```
          4    ..................  1
        2   6    ................  0
      7   2   3    ..............  1
    1   2   1   2    ............  0
  1   5   3   3   3    ..........  1
```

La nostra pallina parte dal premio collocato nel vertice in alto del triangolo (di valore 4), e scende scegliendo, riga dopo riga, se rimbalzare verso sinistra o verso destra ad ogni piolo colpito. Nel passare da una riga alla successiva, non essendo presente in essa un piolo verticalmente allineato a quello in cui ci troviamo, dovremo portarci verso sinistra oppure verso destra, raggiungendo il primo piolo della riga successiva che si incontra in tale direzione. In questo modo, di ogni riga del triangolo verrà colpito precisamente un piolo (ed acquisito il relativo premio), ed il percorso compiuto ha per valore la somma dei valori dei premi acquisiti. Di fatto il percorso compiuto trova codifica in una stringa sull'alfabeto $\{L, R\}$ (sinistra/destra) che specifica le scelte compiute come a partire dal vertice in alto. Dove n è il numero di righe del triangolo, tale stringa è lunga $n - 1$. Ma si noti la colonna verticale di bit disposta sulla destra del triangolo. Quando il bit è 1, stà a noi decidere se portarci a sinistra oppure a destra. Quando il bit è 0 è la casa che farà il suo gioco, cercando di minimizzare il valore totale del percorso e il conseguente esborso in premi. Ad esempio, dal 4 disposto al vertice spetta a noi decidere se portarci verso destra (sul 6) oppure verso sinistra (sul 2), ma la prossima scelta spetterà alla casa. In questo caso, le scelte saranno alternate, ma in generale è questo vettore di bit che specifia a chi spetta ciascuna scelta.

La sfida è massimizzare il valore del percorso che verrà a prodursi nel gioco, seguendo il criterio del caso peggiore, poichè siamo consapevoli che la casa non intende regalarci proprio nulla ed è consapevole che anche noi siamo altrettanto attrezzati.

Dopo aver calcolato il valore del gioco, sarai chiamato a svolgere interattivamente una partita.

NOTA BENE

Nel caso di problemi interattivi come questo è assolutamente importante nel tuo programma fare flush ogni qualvolta si scrive qualcosa

Input

La prima riga contiene T, il numero di testcase da risolvere. Seguono T istanze del problema. Ogni istanza si compone del seguente preambolo: una prima riga fornisce n, il numero di righe del triangolo. Seguono le n righe del triangolo, puoi assumere che esse siano tutte allineate a sinistra. L'ultima riga del preambolo riporta, separati da spazio, gli $n-1$ bit che specificano il giocatore di turno su ciascuna riga, partendo dalla prima e con esclusione dell'ultima dove la pallina si arresta. Poi ha luogo la partita, dove le mosse dei giocatori si avvicendano rispettando i turni. In input riceverai solo le mosse dell'avversario, quando il turno spetta a lui. Per specificare ogni sua mossa, il tuo avversario sul server scrive 'R' oppure 'L' a seconda se intende far scivolare la pallina verso destra oppure verso sinistra.

Output

L'output deve contenere varie righe per ogni testcase. Subito dopo aver ricevuto la descrizione del gioco, scrivi su una prima riga il valore del gioco ricevuto. Poi ha luogo la partita, dove le mosse dei giocatori si avvicendano rispettando i turni. In output dovrai scrivere le tue mosse solo quando è il tuo turno. Per specificare ogni tua mossa, scrivi 'R' oppure 'L' a seconda se intendi far scivolare la pallina verso destra oppure verso sinistra, sapendo che il tuo avversario sul server adopera il tuo stesso linguaggio. Al termine della partita, per completare correttamente il testcase, scrivi due ulteriori righe di reportistica: la prima conterrà la codifica del cammino che ha avuto luogo, la seconda ne conterrà il valore.

Esempio

Input

```
3
1
7

5
4
2 6
7 2 3
1 2 1 2
1 5 3 3 3
1 0 1 0
L
R
5
4
2 6
7 2 3
1 2 1 2
1 5 3 3 3
1 0 1 0
L
L
```

Spiegazione: due testcase, il primo dei quali è un triangolo di una sola riga (e quindi di un solo valore intero, il 7). Il secondo triangolo ha invece 5 righe e così la seconda riga che descrive il testcase consta di quattro bit (nel caso del primo testcase quella riga era vuota). Questi quattro bit specificano che i giocatori sul server e sul client si alternano, la prima mossa spetta al giocatore sul server.

Output

```
7

7
17
R
L
RLLR
17
17
R
L
RLLL
```

```
19
```

Spiegazione: nel secondo tescase, il valore del gioco può essere inteso col seguente ragionamento: la prima scelta spetta a te, e ti conviene portarti sul 6, la casa ti porterà quindi sul 2 della terza riga, e tu a quel punto opterai per andare a sinistra e prenderti il 2 anche della quarta riga. L'ultima scelta della casa è di andare a destra, per consegnarti un 3 piuttosto che un 5. Il valore complessivo che possiamo garantire di raccogliere nel caso peggiore sarà $4+6+2+2+3 = 17$.

Input and Output interleaved

Conviene specificare anche l'ordine rigido in cui sono state scambiate le righe per rispettare il protocollo (ricordiamo l'importanza di fare flush ogni qual volta il protocollo richiede di passare la palla). Per specificare il mittente, inseriamo i caratteri '>' e '<' all'inizio di ciascuna riga.

```
>3
>1
>7
>
<7
<
<7
>5
>4
>2 6
>7 2 3
>1 2 1 2
>1 5 3 3 3
>1 0 1 0
<17
<R
>L
<L
>R
<RLLR
<17
>5
>4
>2 6
>7 2 3
>1 2 1 2
>1 5 3 3 3
>1 0 1 0
<17
<R
>L
```

```
<L
>L
<RLLL
<19
```

Assunzioni

Per il subtasking sono previste le seguenti `size`, dove il default è `big` che include anche i testcase `medium`, `small` e `tiny`:

- `tiny`: $n \leq 7$
- `small`: $n \leq 10$
- `medium`: $n \leq 28$
- `big`: $n \leq 40$

Il tempo limite per testcase è di 1 secondo.