Storie di formiche (`formiche` [106 punti])

Delle formiche marciano su un tavolo orizzontale alla velocità di un'unità di spazio per unità di tempo. Non appena fuoriescono dal bordo sinistro (alla coordinata 0) o destro (alla coordinata m) del tavolo precipitano sul pavimento. Quando una formica si scontra con un'altra formica inverte istantaneamente la propria direzione di marcia. Di ciascuna formica $i = 0, \ldots, n-1$ è nota la coordinata iniziale $x_i \in \mathbb{N}$ sul tavolo e la direzione iniziale di marcia $d_i \in \{L, X, R\}$; quì L significa "verso sinistra", R stà per "verso destra", e X indica che la formica è in realtà morta ed immobile sul tavolo (le altre formiche invertono comunque la propria direzione di marcia quando ci vanno a sbattere).

Con quanta efficienza sai rispondere alle seguenti domande?

[**caduti**]: quali formiche cadono infine dal tavolo e quali formiche non cadono mai?

[**destra-sinistra**]: quali formiche cadono dal bordo destro e quali dal bordo sinistro?

[**inizio**]: in quale istante avviene la prima caduta?

[**fine**]: in quale istante avviene l'ultima caduta?

[**biografia**]: quale è l'istante di caduta di ciascuna formica?

Affidiamoci ora alle suggestioni ricercando un linguaggio evocativo: "Il tempo scorre continuo e le formiche sono così piccole che dovresti pensare a loro come a dei punti immateriali che si muovono con velocità costante, invertendo la loro velocità in un istante. Lo stesso tempo che due formiche sono in contatto quando si scontrano è immateriale e non ha durata alcuna."

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 è descritta in tre righe: la prima contiene i due numeri $n, m \in \mathbb{N}$, la seconda contiene gli n caratteri $d_i \in \{L, X, R\}$, $i = 0, \ldots, n-1$, riportati nell'ordine e separati da spazio, la terza contiene le n coordinata iniziali $x_i, i = 0, \ldots, n-1$, nell'ordine e separate da spazio. Si assume che esse siano n numeri *interi* e *pari*, *tutti diversi* tra di loro, e *strettamente contenuti* nell'intervallo aperto $(0, m)$.

Output

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

[**goal 1+2**]: la prima riga contiene, separati da spazio, gli n caratteri $c_i \in \{L, T, R\}$ dove $c_i = T$ dice che la formica rimarrà per sempre sul tavolo mentre $c_i \in \{L, R\}$ indica che la formica i infine cade dal tavolo (più precisamente, per il `goal 3`, dal bordo sinistro se $c_i = L$ o da quello destro se $c_i = R$).

[**goal 3**]: la seconda riga riporta un unico numero intero che specifica a quale istante si verifica la prima caduta di una formica.

[**goal 4**]: la terza riga riporta un unico numero intero che specifica a quale istante si verifica l'ultima caduta di una formica.

[**goal 5**]: la quarta riga contiene n numeri interi t_i separati da spazi: $t_i = -1$ se la formica i non cade mai, altrimenti t_i è l'istante della sua caduta.

Oltre alle dimensioni delle istanze, ogni subtask precisa quanti punti competono ai vari goal. Per ciascuna istanza di quel subtask si otterranno tutti i punti dei goal evasi con risposta è corretta (purchè si rispetti almeno il formato in tutte le righe di output, incluse quelle che competono agli altri goal – altrimenti salta il protocollo di comunicazione tra il tuo programma risolutore e il server).

Esempio di Input/Output

Input da `stdin`

```
-start-
2
5 18
R R L L L
2 6 8 14 16
-more-
```

```
6 16
R R X R L R
2 4 8 10 12 14

 -end-
```

Output su `stdout`

```
L L L R R
8
16
8 14 16 16 12
L L T R R R
2
14
12 14 -1 12 6 2
```

Spiegazione: Nel primo di 2 testcase abbiamo 5 formiche che camminano su di un tavolo di lunghezza 18. Al tempo 0 la formica A ha coordinata 2, B è in 6, C in 8, D in 14, ed E in 16. Le coordinate e direzioni nei vari tempi (quelli possiamo discretizzarli, gli istanti in cui succede qualcosa saranno sempre degli interi) sono le sequenti:

```
     000 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018
              >               >       <                       <       <
t=0:          A               B       C                       D       E

                  >               C                       <       <
t=1:              A               B                       D       E

                      >       <       >               <       <
t=2:                  A       B       C               D       E

                          B               >       <       <
t=3:                      A               C       D       E

                      <       >               D       <
t=4:                  A       B               C       E

                  <               >       <       E
t=5:              A               B       C       D

              <                       C       <       >
t=6:          A                       B       D       E

          <                       <       D               >
t=7:      A                       B       C               E

      <                       <       <       >               >
t=8:  A                       B       C       D               E
```

E da questa rappresentazione, senza bisogno di completarla, si possono già verificare i valori in output e comprendere diverse cose.

Assunzioni

1. $2 < n < 10^6$
2. x_i è un numero intero pari, per ogni $i = 0, 1, ..., n-1$
3. $0 < x_i < x_{i+1} < m \leq 10^9$ per ogni $i = 0, 1, ..., n-2$
4. in tutte le istanze che verranno fornite cadrà sempre almeno una formica sia dal bordo sinistro che da quello destro

Subtask

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

I testcase sono raggruppati nei seguenti subtask.

1. **[4 pts← 2 istanze da 0 + 1 + 0 + 0 + 1 punti] `esempi_testo`:** i due esempi del testo
2. **[15 pts← 5 istanze da 0 + 0 + 1 + 1 + 1 punti] `tiny`:** $n \leq 10, m \leq 50$, nessuna formica morta
3. **[10 pts← 5 istanze da 0 + 1 + 0 + 0 + 1 punti] `small`:** $n \leq 20, m \leq 50000$, nessuna formica morta
4. **[16 pts← 8 istanze da 0 + 1 + 0 + 0 + 1 punti] `medium`:** $n \leq 100, m \leq 200$, nessuna formica morta
5. **[16 pts← 8 istanze da 0 + 1 + 0 + 0 + 1 punti] `big`:** $n \leq 500, m \leq 1,000,000$, nessuna formica morta
6. **[15 pts← 5 istanze da 1 + 0 + 0 + 0 + 2 punti] `big_with_death`:** $n \leq 500, m \leq 1,000,000$, max 3 formiche morte
7. **[30 pts←15 istanze da 0 + 0 + 0 + 0 + 2 punti] `large`:** $n \leq 50,000, m \leq 1,000,000$, nessuna formica morta

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
       formiche  -- python my_solution.py
```

vengono valutati, nell'ordine, i subtask:

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

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

Consigli

1. Questo problema si presta naturalmente ad una simulazione, ma c'è anche un'algoritmo $O(n)$.
2. La simulazione otterrà solo parte dei punti poichè impiega $\Theta(n^2 \log n)$ tempo (oppure $\Theta(nm)$, ossia pseudo-polinomiale nell'input, se fai scorrere il tempo di un'unità alla volta piuttosto che usare una priority-queue in una event driven simulation). Inoltre codificare la simulazione, anche solo la $\Theta(nm)$ che è più facile di non poco, resta comunque più oneroso che non codificare una soluzione $\Theta(n)$ cui puoi pervenire solo dopo aver analizzato e compreso il problema. Di converso, per giungere all'algoritmo lineare devi entrare nel problema ossia devi darti il tempo per sperimentare con esso con curiosità. Vedi tu come ti conviene gestirti in funzione dei tuoi obiettivi col corso, della tua strategia di singolo appello, e del contingente. (Se hai poco tempo e punti all'accumulo di quantità anche limitate ma non trascurabili di punti considera la simulazione pseudopolinomiale.)
3. In ogni caso ti conviene affrontare il problema per gradi, sia nelle comprensione che nella codifica. In particolare, nota che ben pochi subtask presentano formiche morte. Ti conviene approfittare di ciò (si è inteso premiare e promuovere l'approccio solido e in ultima più conveniente).

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

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