Poprawne dwunawiasowanie **(dwunawiasowanie)**

Limit pamięci: 32 MB Limit czasu: 1.00 s

Poprawnym dwunawiasowaniem nazwiemy każdy ciąg złożony ze znaków `([)]`, który można otrzymać z rekurencyjnej zależności:

- Ciąg pusty jest poprawnym dwunawiasowaniem
- Jeśli `P` jest poprawnym dwunawiasowaniem, to `[P]`, oraz `(P)` są poprawnymi dwunawiasowaniami
- Jeśli `P` i `Q` są poprawnymi dwunawiasowaniami, to `PQ` jest poprawnym dwunawiasowaniem

Przykładowo, `[()[]]` jest poprawnym dwunawiasowaniem, ale `)(]`, lub `[(]` nie są.

Napisz program, który wczyta ciąg nawiasów oraz zapytania o poprawność dwunawiasowania wybranych spójnych podsłów ciągu, wyznaczy dla każdego zapytania czy dwunawiasowanie jest poprawne i wypisze wyniki na standardowe wyjście.

Wejście

W pierwszym wierszu wejścia znajdują się jedna liczba N, oznaczająca długość ciągu.
W drugim wierszu wejścia znajduje się ciąg nawiasów o długości N.
W trzecim wierszu wejścia znajdują się jedna liczba Q, oznaczająca ilość zapytań.
W kolejnych Q wierszach znajduje się opis kolejnych zapytań, po jednym w wierszu. Opis każdego zapytania składa się z dwóch liczb naturalnych L_i, R_i, oddzielonych pojedynczym odstępem. Określają one zapytanie o poprawność zadanego spójnego podciągu dwunawiasowania od L_i-tego znaku do R_i-tego znaku włącznie.

Wyjście

Twój program powinien wypisać na wyjście dokładnie Q wierszy. W i-tym wierszu powinna się znaleźć odpowiedź dla i-tego zapytania. Odpowiedź dla każdego zapytania to jedno słowo `TAK`, jeśli dwunawiasowanie jest poprawne lub `NIE` w przeciwnym przypadku.

Ograniczenia

$1 \leq N, Q \leq 1\ 000\ 000$, $1 \leq L_i \leq R_i \leq N$.
W testach wartych 30% wszystkich punktów zachodzi $1 \leq N, Q \leq 2\ 000$.
W testach wartych 20% wszystkich punktów ciąg składa się tylko ze znaków `()`.

Przykład

Wejście	**Wyjście**
5 [()][5 1 4 2 3 1 2 1 3 1 5	TAK TAK NIE NIE NIE

Wejście	**Wyjście**
4 [(]) 1 1 4	NIE

Kółko i Krzyżyk (duże) **(kolko-i-krzyzyk-big)**

Limit pamięci: 32 MB Limit czasu: 1.00 s

W Las Vegas odbywają się zawody najniższej klasy. O tytuł mistrza Zbąszynka w kółko i krzyżyk walczą najwybitniejsi stratedzy ze Skellige: Magnus Carlsberg, oraz Włodzimierz Biały. Tak się jednak składa, że włodarz gali, który jest bułgarskim samurajem, obstawił cały majątek na wygraną jednego z zawodników. Z obawy że partia może zakończyć się remisem postanowił zmodyfikować zasady. W tej partii gra toczyć się będzie na prostokątnej planszy rozmiaru $N \times M$. Zwycięży jednak standardowo ten, kto ułoży trzy ze swoich figur w wierszu, kolumnie, bądź na ukos. Rozgrywkę zaczyna Magnus Carlsberg. Bułgarski samuraj zastanawia się jaki będzie wynik dla danej konfiguracji, przy założeniu że obaj gracze grają optymalnie.

Wejście

W pierwszym i jedynym wierszu wejścia znajdują się dwie liczby całkowite N, oraz M.

Wyjście

W pierwszym i jedynym wierszy wyjścia znajdują się napis “Magnus Carlsberg”, jeśli zwycięży Magnus Carlsberg, “Włodzimierz Biały”, jeśli zwycięży Włodzimierz Biały, oraz “Remis” w przypadku remisu.

Ograniczenia

$1 \leq N, M \leq 2500$.

Przykład

Wejście	Wyjście	Wyjaśnienie
3 3	Remis	Znany przypadek, w którym żaden z graczy nie ma strategii wygrywającej

Mini Saper **(mini-saper)**

Limit pamięci: 256 MB Limit czasu: 2.00 s

To zadanie jest interaktywne.

Pewnie wielu z was słyszało o grze Saper. W zadaniu mamy doczynienia z uproszczoną wersją tej gry. Twoim zadaniem jest oznakowanie pojedynczej bomby, która znajduje się na planszy o wymiarach 3 na 3. Aby to uczynić należy odkrywać pola niezawierające bomby. Każde takie pole zawiera informację o tym czy pole zawierające bombę znajduje się w otoczeniu odkrytego pola. Powiemy, że pole jest w otoczeniu innego pola, jeżeli stykają się one bokiem lub wierzchołkiem. Odkrycie bomby skutkuje porażką. Aby umożliwić Tobie skuteczne oznakowanie bomby, masz zagwarantowane, że bomba nie znajduje się w polu $(1, 1)$.

Protokół interakcji

Do komunikacji z programem sprawdzającym należy używać poniższych zapytań:

- `odkryj i j` – odkrywa pole w i–tym wierszu i j–tej kolumnie. Wiersze i kolumny numerujemy od jedynki. W przypadku próby odkrycia pola poza planszą; pola, które zostało już wcześniej odkryte lub pola, które zawiera bombę zostanie wypisane na wejście `-1`. W tym przypadku należy zakończyć działanie programu. W przeciwnym wypadku na wejście zostanie wypisane `1`, jeżeli w otoczeniu pola (i, j) znajduje się bomba lub `0` w przeciwnym wypadku.
- `bomba i j` – oznakowuje pole w i–tym wierszu i j–tej kolumnie. W przypadku poprawnego oznakowania test zostanie zaliczony, a w przeciwnym, zostanie zwrócony odpowiedni werdykt informujący o niepoprawnym oznaczeniu pola. W obu przypadkach należy zakończyć dalszą interakcję z programem.

Należy pamiętać o opróżnianiu bufora wypisywania po każdym zapytaniu. Aby to uczynić należy wykonać `cout.flush();` lub `cout << endl` jeżeli używamy cin/cout w C++, `fflush(stdout)` dla printf/scanf w C++, `sys.stdout.flush()` w Pythonie oraz `System.out.flush()` w Javie.

Przykładowa interakcja

Wejście	Wyjście
	`odkryj 1 1`
`1`	
	`odkryj 3 3`
`1`	
	`bomba 2 2`

Wyjaśnienie przykładu: Zapytanie o odkrycie pola $(1, 1)$, dało nam informację, że bomba znajduje się na którymś z pól: $(1, 2), (2, 2), (2, 1)$. Następne zapytanie ujawnia, że bomba musi się na którymś z pól: $(3, 2), (2, 2), (2, 3)$. Zatem wiemy, że bomba znajduje się na polu $(2, 2)$. Po zapytaniu `bomba 2 2` należy zakończyć interakcję z programem.