Рицарски турнир

За сватбената си церемония с Беатриче Д'Есте през 1491 г., Дукът на Милано Людовик Сфорца поканил Леонардо да организира сватбеното тържество, като включи и голям рицарски турнир, който да продължи три дни и три нощи. Обаче най-известният рицар закъснява ...

Турнир

В началото на всеки рицарски турнир всичките N участници се подреждат в редица и заемат последователно позиции с номера от 0, 1, ..., N - 1. На всеки кръг водещият турнира (а това е самият Леонардо) обявява две позиции S и E (където $0 \leq S < E \leq N - 1$). Всички рицари, намиращи се на позиции от S до E (включително) се състезават помежду си: само победителят продължава участието си и се връща на своята позиция, а останалите рицари напускат турнира. След това рицарите се преместват към началото на редицата, така че да няма празни позиции. Следва втори тур и т.н., докато остане само един рицар.

Леонардо знае, че рицарите имат различна сила. Силата се измерва с някакво число от 0 (най-слаб) до N - 1 (най-силен) и няма двама рицари с еднаква сила. Леонардо знае предварително списъка от команди (C на брой), определящи отделните кръгове на турнира. Абсолютно сигурно е, че винаги побеждава най-силният рицар от групата.

Закъснелият рицар

N - 1 от всичките N рицари са вече подредени в редицата, само най-известният рицар все още липсва. Този рицар има ранг R и пристига малко по-късно. За да бъде по-забавно, Леонардо иска да използва популярността на този рицар и избира за него такава позиция в редицата, при която броят на кръговете, в които рицарят побеждава е максимално възможен. Да отбележим, че не се интересуваме от кръгове, в които най-известният рицар не участва, а само от такива кръгове, в които рицарят участва и побеждава.

Пример

За N = 5 рицари, има N - 1 рицари, които вече са подредени и те имат сила [1, 0, 2, 4], съответно. Следователно закъснелият рицар има сила R = 3. За C = 3 кръга, водещият възнамерява да извика следните (S, E)-двойки в този ред: (1, 3), (0, 1), (0, 1).

Ако Леонардо вмъкне рицаря на първа позиция, ранговете ще бъдат [3, 1, 0, 2, 4]. В първия кръг участват рицарите на позиции 1, 2 и 3, с рангове 1, 0, 2, съответно. Победителят е с ранг 2. Новата редица е: [3, 2, 4]. В следващият кръг, който е на позиции 0 и 1, побеждава рицарят с ранг R = 3, като редицата става [3, 4]. В последни кръг (на

позиции 0, 1) побеждава 4. По такъв начин закъснелият рицар е победител само в един кръг (втория).

Ако вместо това Леонардо вмъкне закъснелия рицар между двамата рицари с ранг 1 о 0, редицата ще изглежда така: [1, 3, 0, 2, 4]. В първия кръг участват 3, 0, 2 и рицарят с ранг R = 3 побеждава. Линията сега е [1, 3, 4] и на следващия кръг (1 срещу 3) рицарят с ранг R = 3 отново побеждава. Накрая линията е [3, 4], където 4 побеждава. Така закъснелият рицар ще победи в два кръга.Това всъщност е най-добрата възможна позиция, тък като няма друг начин закъснелият рицар да победи в повече от два кръга.

Задача

Напишете програма, която намира най-добрата позиция за закъснелия рицар, така че броят на кръговете, в които той побеждава, да бъде максимален.Трябва да реализирате функция, с име `GetBestPosition`(N, C, R, K, S, E), където:

- N е броя на рицарите;
- C е броя на кръговете, обявени от водещия турнира ($1 \leq C \leq N - 1$);
- R е рангът на закъснелия рицар; ранговете на всичките рицари (на тези, които са вече в редицата и на закъснелия рицар) са различни числа от интервала 0, …, N - 1 и рангър R на закъснелия рицар е даден експлицитно, макар че може да буде получен от останалите данни;
- K е масив от N - 1 цели числа, представляващ ранговете на N - 1 рицари, които са вече на стартовата линия;
- S и E са два масива от по C елемента: за всяко i от 0 до C - 1, включително, (i + 1)-първият кръг, обявен от водещия турнира включва всички рицари от позиция S[i] до позиция E[i], включително. За всяко i е изпълнено S[i] < E[i].

Обръщенията към тази програма са коректни: числото E[i] е по-малко от броя на оставащите рицари за (i + 1)-вия кръг и след изпълнението на всичките C команди ще има точно един останал рицар.

`GetBestPosition`(N, C, R, K, S, E) трябва да връща най-добрата позиция P, където Леонардо трябва да вмъкне закъсняващия рицар ($0 \leq P \leq N - 1$). В случай, че има няколко екривалентни позиции изведете най-малката от тях. Позицията P е 0-базирана позиция на закъснелия рицар в резултатната линия. С дриги думи, P е броят на другите рицари, които са преди закъсняващия рицар в оптималното решение. Специално, P = 0 означава, че закъснелият рицар трябва да бъде първи в редицата, докато P = N - 1 означава, че той трябва да е последен.

Подзадача 1 [17 точки]

Ще бъде изпълнено $N \leq 500$.

Подзадача 2 [32 точки]

Ще бъде изпълнено $N \leq 5\,000$.

Подзадача 3 [51 точки]

Ще бъде изпълнено $N \leq 100\,000$.

Подробности по реализацията

Трябва да изпратите само един файл, с име `tournament.c`, `tournament.cpp` или `tournament.pas`. Този файл трябва да съдържа функциите, описани по-горе, използвайки следните прототипи.

Програми на C/C++

```
int GetBestPosition(int N, int C, int R, int *K, int *S, int *E);
```

Програми на Pascal

```
function GetBestPosition(N, C, R : LongInt; var K, S, E : array of LongInt) : LongInt;
```

Тези функции трябва да работят според описанието. Може да реализирате и други функции за вътрешна употреба. Вашите програми не трябва да взаимодействат по никакъв начин със стандартния вход или стандартния изход, нито пък с някакви други файлове.

Примерен грейдър

Примерният грейдър ще очаква вход в следния формат:

- ред 1: N, C, R;
- ред 2, …, N: K[i];
- редове N + 1, …, N + C: S[i], E[i].

Ограничения по време и памет

- Ограничение по време: 1 секунда.
- Ограничение по памет: 256 MiB.