馬背射擊比賽

1491年, Milan Lodovico Sforza 公爵就他和 Beatrice d'Este 之間的婚禮要求 Leonardo 來籌備婚禮相關慶祝活動，包括一個偉大的馬背射擊角逐比賽， 該比賽將整整進行三天。但當中最熱門的騎士卻遲到了.....

角逐比賽

在這場角逐比賽中，N 位騎士首先將會排在一條開始線上，他們的位置將順序被編號為 0 到 N-1。比賽的主持通過決定兩個位置 S 和 E ($0 \le S < E \le N - 1$) 來決定一輪比賽。在該輪比賽中, 所有位置在 S 和 E 之間 (包括 S 和 N 在內) 的所有騎士都需要參加角逐比賽。比賽中勝出的騎士可以返回他原來在開始線上的位置， 而落敗的騎士則要離開賽場。留在賽場的騎士將會向開始線的首位靠攏並保持原來的相互之間的相對位置順序， 這些騎士將被重新編號為 0 至 N - （E - S） - 1。然後比賽主持將再決定另一輪比賽的騎士。這樣一直進行下去直至場上只餘下一位騎士。

Leonardo 知道所有的騎士有不同的強壯度，並以不同的等級表示之, 從 0（最弱）到 N - 1（最強）。他也準確知道對於任一輪比賽 C， 主持人將會給出的所有命令，畢景他是 Leonardo ... 他亦知在這些比賽中, 每輪等級最高的騎士將會勝出。

遲來的騎士

N 位騎士中的 N - 1 位已經排好在開始線上，只是最熱門的那位騎士還未到。這位等級為 R 的騎士來得有些遲。為了增加娛樂性，Leonardo 希望利用這位遲來的騎士的知名度，並為他選擇一個位置，使得這位遲到的騎士參賽時能取得最多的獲勝輪數。請注意，我們對這位騎士不參與的輪數不感興趣，只是計算他參與並獲勝的輪數。

樣例

對於 N = 5 位騎士而言, N - 1 位騎士已在開始線上排好隊, 他們的等級分別順次為 [1, 0, 2, 4]。所以遲來的騎士的等級為 R = 3。等於輪數 C = 3 輪而言, 比賽主持將會決定的 (S, E) 位置編號分別順次為：(1, 3), (0, 1), (0, 1)。

若 Leonardo 將遲到的騎士編在第一位上, 則在開始線上的騎士的等級將為 [3, 1, 0, 2, 4]。這樣, 第一輪比賽將包括騎士編號 1, 2, 3 則等級順次為 1, 0, 2 所以等級為 2 的騎士將會勝出。比賽後新組成的開始線上的騎士(等級)順次為 [3, 2, 4]。下一輪比賽第級為 3 的騎士(位置0)面對等級為2的騎士(位置1), 且第級為 R=3 的騎士將會勝出, 餘下的騎士(等級)順次為[3, 4]。而最後一輪比賽中, 位置 0 的騎士將對位置 1 的騎士, 當然等級為4的那位騎士將會勝出。這樣一來, 遲來的騎士就只能在一輪比賽(第二輪比賽)中獲勝。

但若 Leonardo 將遲來的騎士加在等級為 1 及 0 的騎士之間, 開始線上的騎士第級順序將變為[1, 3, 0, 2, 4]。這時, 第一輪比賽將包括等級為 3, 0, 2 的三位騎士, 而等級為 R=3 的騎士

將勝出。開始線上的騎士將變為 [1, 3, 4], 而下一輪比賽中, 等級為 1 的騎士將面對等級為 3 的騎士, 因而等級為 R=3 的騎士將再次勝出。而最後一場比賽中, 開始線上的騎士將餘下 [3, 4], 當然等級為4的騎士將勝出。這樣一來, 遲來的那位騎士將勝出兩場。事實上, 這是最好的安排, 因為沒有任何其他可能的安排會使這位遲來的騎士勝出多過兩場。

說明

你的任務是編寫一程式以選擇一個最好的位置給這位遲到的騎士使他勝出的輪數為最多, 正如 Leonardo 所願。實際地, 你需要編寫一子程式名為 `GetBestPosition`(N, C, R, K, S, E), 其中:

- N 是騎士的數目;
- C 是比賽主持要負責的輪數 ($1 \le C \le N - 1$);
- R 是遲來的騎士的等級; 所有騎士的等級(包括那些已在起點線上及那位遲來的騎士在內)均為由 0, ..., N-1 之間選出並不重複的整數。且遲來的騎士的等級 R 將明確地提供給你雖然它是可以由其他騎士的等級中推算出來的。

K 是一個含有 N - 1 個整數的陣列, 它們順次表示著這 N - 1 位已在開始線上排好隊的騎士的等級;

- S 及 E 是兩個大小為 C 的陣列: 對於每個在 0 和 C-1 之間的整數 i (包含 0 及 C-1 在內), 比賽主持在第 i + 1 輪比賽中將會叫出位置在 S[i] 至 E[i] 的所有騎士參加角逐。你可以假設對所有 i 而言, S[i] < E[i]。

所有對這些子程式的乎叫都會是合法的: E[i] 會是少於第 i+1 輪比賽中餘下的騎士數目, 而當最後一輪比賽指今給出後,場上應只餘下一位騎士。

`GetBestPosition`(N, C, R, K, S, E) 必要返回位置值 P, 該值代表 Leonardo 應該把那位遲到的騎士安排的最好位置。如果有若干個置位都是用樣的最好的話, *則請輸出最少的一個*。(位置 P 是以 0 為開始的, 它應代表將遲到的騎士安置後, 他在開始線上的位置。換言之, P 是在最好的解中, 排在遲到的騎士前面的騎士數目。如果 P=0, 則代表遲到的騎士是排在第一位, 而若 P=N-1則代表遲到的騎士是排在最後一位)。

子任務 1 [17 分]

你可以假設 $N \le 500$.

子任務 2 [32 分]

你可以假設 $N \le 5\,000$.

子任務 3 [51 分]

你可以假設 $N \le 100\,000$.

編程實現說明

你必須只提交一程式檔案 `tournament.c`, `tournament.cpp` 或 `tournament.pas`。這檔案必須以上述的相關函數編寫並符合下面的說明。

C/C++ 程式

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

Pascal 程式

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

這些子程式一定要根据上述的特點來編寫。當然你亦可自由地編寫其他在計算過程中需要的子程式。你所編寫的程式不能與標準輸入輸出有任何的直接互動。亦不能使用任何其他的檔案。

樣例 graders

樣例 grader 將提供需要以下輸入格式的評測環境:

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

時間及記憶體限制

- 時間限制: 1 秒
- 記憶體限制: 256 MiB.