Turnierwettkampf

Der Herzog von Mailand, Ludovico Sforza, bat Leonardo 1491 die Feier seiner Hochzeit mit Beatrice d'Este zu organisieren. Dabei sollte ein großartiger, dreitägiger Turnierwettkampf stattfinden. Aber der berühmteste Ritter kommt zu spät...

Turnier

In einem Turnierwettkampf werden die N Ritter zuerst in einer Reihe angeordnet, wobei ihre Positionen von 0 bis N-1 der Reihe nach durchnummeriert sind. Der Turniermeister legt durch den Aufruf von zwei Positionen S und E (wo $0 \leq S < E \leq N - 1$) eine *Runde* fest. Alle Ritter, deren Positionen zwischen S und E (inklusive) liegen, nehmen teil: Der Gewinner setzt das Turnier fort und geht auf seinen ursprünglichen Platz in der Reihe zurück, die Verlierer sind aus dem Spiel und verlassen das Feld. Danach rücken die verbleibenden Ritter in Richtung Beginn der Reihe zusammen. Dabei wird die relative Ordnung beibehalten, sodass die Positionen der Ritter im Bereich von 0 bis N - (E - S) - 1 liegen. Der Turniermeister ruft eine neue Runde aus und wiederholt diesen Vorgang, bis nur mehr ein Ritter übrig bleibt.

Leonardo weiß, dass alle Ritter unterschiedliche Stärken haben, welche durch ein Ranking von 0 (schwächster Ritter) bis N-1 (stärkster Ritter) gegeben sind. Keine zwei Ritter haben gleiches Ranking. Er kennt auch genau die Kommandos, welche der Turniermeister für die C Runden ausrufen wird. Leonardo ist sich sicher, dass jede dieser Runden der Ritter mit dem höchsten Ranking gewinnen wird.

Verspäteter Ritter

N-1 der N Ritter sind bereits in der Reihe aufgestellt, nur der populärste Ritter fehlt. Dieser Ritter hat das Ranking R und trifft ein bisschen später ein. Um möglichst große Unterhaltung zu bieten, möchte Leonardo seine Popularität ausnutzen; er wählt für den verspäteten Ritter eine Position in der Reihe, welche die Anzahl der Runden, die dieser gewinnen wird, maximiert. Beachte, dass wir an Runden, an denen der verspätete Ritter nicht teilnimmt, nicht interessiert sind, sondern nur an Runden, an denen er teilnimmt und gewinnt.

Beispiel

Für N = 5 Ritter sind die N-1 Ritter bereits in einer Reihe mit den Rankings [1, 0, 2, 4] aufgestellt. Konsequenterweise hat der verspätete Ritter das Ranking R = 3. Für C = 3 Runden hat der Turniermeister vor, die (S, E) Positionen pro Runde in folgender Reihenfolge auszurufen: (1, 3), (0, 1), (0, 1).

Wenn Leonardo den verspäteten Ritter an der ersten Position einfügt, würden die Ritter in dieser

Rankingreihe stehen: [3, 1, 0, 2, 4]. Die erste Runde betrifft die Ritter (an den Positionen 1, 2, 3) mit den Rankings 1, 0, 2; der Ritter mit dem Ranking 2 gewinnt. Die neue Reihe ist dann [3, 2, 4]. Die nächste Runde ist 3 gegen 2 (an den Positionen 0, 1) und der Ritter mit dem Ranking R = 3 gewinnt. Die Reihe [3, 4] bleibt übrig. Die finale Runde (mit den Positionen 0, 1) hat 4 als Gewinner. Also gewinnt der verspätete Ritter nur eine Runde (die zweite).

Anderenfalls, wenn Leonardo den verspäteten Ritter zwischen den beiden Rittern mit Ranking 1 und 0 einfügt, sieht die Reihe folgendermaßen aus: [1, 3, 0, 2, 4]. Dieses Mal sind die Ritter mit den Rankings 3, 0, 2 in der ersten Runde beteiligt und der Ritter mit dem Ranking R = 3 gewinnt. Die neue Reihe ist [1, 3, 4]. In der nächsten Runde (Ranking 1 gegen 3) gewinnt wieder der Ritter mit dem Ranking R = 3. Die letzte Reihe ist [3, 4], wobei 4 gewinnt. Also gewinnt der verspätete Ritter zwei Runden: Dies ist die beste mögliche Platzierung, denn es gibt keine Möglichkeit für den verspäteten Ritter, mehr als zwei Runden zu gewinnen.

Aufgabe

Schreibe ein Programm, welches die beste Position für den verspäteten Ritter wählt, sodass er maximal viele Runden gewinnt. Insbesondere musst du ein Unterprogramm namens `GetBestPosition`(N, C, R, K, S, E) implementieren, wobei:

- N ist die Anzahl der Ritter;
- C ist die Anzahl von Runden, die vom Turniermeister ausgerufen werden ($1 \leq C \leq N - 1$);
- R ist das Ranking des verspäteten Ritters; die Rankings aller Rittern sind verschieden (sowohl die der bereits aufgestellten Ritter als auch das des verspäteten Ritters) und liegen im Bereich von 0, ..., N - 1. Das Ranking R des verspäteten Ritters ist explizit gegeben, obwohl es eigentlich hergeleitetet werden kann;
- K ist ein Array von N - 1 Integerwerten, die das Ranking der N - 1 Ritter angeben, die bereits in der Startreihe aufgestellt sind;
- S und E sind zwei Arrays der Größe C: Für jedes i zwischen 0 und C - 1 (inklusive) wird die (i+1)-te Runde vom Turniermeister ausgerufen, wobei die Ritter von den Positionen S[i] bis E[i] (inklusive) beteiligt sind. Du kannst annehmen, dass für jedes i gilt: S[i] < E[i].

Die Aufrufe, die diesem Unterprogramm übergeben werden, sind gültig: wir haben dafür gesorgt, dass E[i] kleiner ist als die laufende Anzahl von verbliebenen Rittern für die (i+1)-te Runde und nach allen C Aufrufen nur mehr exakt ein Ritter übrig bleibt.

`GetBestPosition`(N, C, R, K, S, E) muss die beste Position P zurückgeben, an der Leonardo den verspäteten Ritter positionieren soll ($0 \leq P \leq N - 1$). Wenn es mehrere gleichwertige Positionen gibt, *gib die kleinste Position zurück*. (Die Position P ist die 0-basierte Position des verspäteten Ritters in der resultierenden Reihe. Mit anderen Worten, P ist die Anzahl anderer Ritter, die vor dem verspäteten Ritter in der optimalen Lösung stehen. Insbesondere bedeutet P = 0, dass der verspätete Ritter am Anfang der Reihe steht, und P = N - 1 bedeutet, dass er am Ende der Reihe steht.)

Subtask 1 [17 Punkte]

Du kannst annehmen, dass $N \leq 500$ ist.

Subtask 2 [32 Punkte]

Du kannst annehmen, dass $N \leq 5\,000$ ist.

Subtask 3 [51 Punkte]

Du kannst annehmen, dass $N \leq 100\,000$ ist.

Implementierungsdetails

Du musst genau eine Datei namens `tournament.c`, `tournament.cpp` oder `tournament.pas` einreichen. In dieser Datei ist das oben beschriebene Unterprogramm mit der folgenden Signatur zu implementieren.

C/C++ Programme

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

Pascal Programme

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

Diese Unterprogramme müssen sich wie oben beschrieben verhalten. Natürlich kannst du andere Unterprogramme für interne Verwendung implementieren. Deine Übermittlung darf weder Standard-Input oder -Output noch eine andere Datei benützen.

Beispiel-Grader

Der Beispiel-Grader liest die Eingabe im folgenden Format:

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

Limits

- Zeitlimit: 1 Sekunde.
- Speicherlimit: 256 MiB.