Sortowanie (B)

Limit pamięci: 256 MB Limit czasu: 4.00 s

To zadanie jest interaktywne. Po wypisaniu każdego wiersza należy opróżnić bufor. W C++ możesz użyć `cout << flush`, w Pythonie – `sys.stdout.flush()`. Należy ściśle trzymać się protokołu opisanego w sekcji *Interakcja* – niewczytanie wiadomości wysłanej przez interaktor skutkuje werdyktem *wrong answer* lub innym.

---

3. *Frymbulin włożył do koszyczka z grzybami kilka kawałków rudy żelaza. Kiedy wrócił do domu, zaczął zastanawiać się, do czego mogłoby się przydać jego znalezisko. A dlatego, że to był skrzatek bardzo sprytny, zaraz wpadł mu do głowy pewien pomysł.*
4. *Zbudował prosty piec na węgiel drzewny i wytopił w nim żelazo. Potem wykuł nóż, pokroił grzyby i ugotował z nich pyszną zupę. Mniam mniam.*

Oprócz zupy grzybowej, Frymbulin bardzo lubi sortowanie przez scalanie. W tym celu potrzeba jednak stworzyć procedurę merge. Jako że komputer zbudowany z żelaza ma trochę inne operacje niż zwykły, nie jest to trywialne zadanie. Musisz mu w tym pomóc.

Dane są liczby N, M oraz pewien ukryty ciąg długości $N + M$, składający się z parami różnych liczb. Dodatkowo, wiadomo o nim, że powstał on poprzez *połączenie* (jeden po drugim) *dwóch rosnących ciągów* – pierwszego o długości N i drugiego o długości M.

Należy posortować ten ciąg, używając następujących operacji:

- `?` i j oznacza porównanie dwóch *różnych* elementów na pozycjach i oraz j. W odpowiedzi dostajemy znak < lub >, w zależności od tego, który z elementów jest większy
- `^` i j oznacza odwrócenie fragmentu między pozycjami i oraz j włącznie

Istnieją limity na łączną liczbę operacji oraz na sumę długości odwracanych fragmentów. **Zależą one od podzadania**, podobnie jak limity na N i M.

Interakcja

Interaktor nie jest adaptacyjny – ciągi są ustalone na początku i nie zmieniają się w zależności od zadanych zapytań.

Na początku należy wczytać liczby całkowite N, M, L_{oper}, L_{sum}. Wartości L_{oper} i L_{sum} oznaczają limit na liczbę operacji i sumę długości odwracanych fragmentów.

Następnie należy wykonać ciąg operacji, każdą poprzez wypisanie odpowiedniego wiersza, a następnie **opróżnienie bufora**. Każda operacja zaczyna się od znaku `?` lub `^`.

Jeśli operacja to porównanie, po `?` powinny nastąpić dwa indeksy $1 \leq i \neq j \leq N + M$ porównywanych elementów. Następnie powinien być wczytany znak > lub < – odpowiedź na zapytanie (> jeśli element na pozycji i jest większy, w przeciwnym przypadku <).

Jeśli operacja to odwrócenie, po znaku `^` powinny nastąpić dwa indeksy $1 \leq i \leq j \leq N + M$ – krańce odwracanego przedziału.

Po wykonaniu wszystkich operacji należy wypisać znak `!`, opróżnić bufor i zakończyć działanie programu. Nie należy już nic więcej wypisywać. Jeśli ciąg został posortowany i zmieściliśmy się w limitach, test zostanie zaakceptowany.

Ograniczenia

We wszystkich podzadaniach zachodzi $2 \leq N, M \leq 50\ 000$. Pierwszych N elementów ciągu tworzy ciąg rosnący, podobnie jak ostatnich M elementów.

Podzadania

Poniższa tabelka zawiera ograniczenia na N, M, L_{oper}, L_{sum} w każdym z podzadań.

	N	M	L_{oper}	L_{sum}	**Punkty**
1	50	50	100 000	3 000 000	18
2	500	500	100 000	3 000 000	11
3	500	5 000	100 000	3 000 000	8
4	500	50 000	100 000	3 000 000	24
5	5 000	5 000	50 000	3 000 000	13
6	5 000	5 000	100 000	200 000	14
7	50 000	50 000	300 000	3 000 000	12

Przykładowa interakcja

Wejście	**Wyjście**	**Wyjaśnienie**
`2 3 100000 3000000`		Ciąg to $(1, 5, 2, 3, 4)$
	`? 2 5`	
`>`		$5 > 4$
	`^ 2 5`	Odwracamy fragment $(5, 2, 3, 4)$, ciąg teraz to $(1, 4, 3, 2, 5)$
	`? 3 1`	
`>`		$3 > 1$
	`? 1 4`	
`<`		$1 < 2$
	`^ 2 4`	Odwracamy fragment $(4, 3, 2)$, ciąg teraz to $(1, 2, 3, 4, 5)$
	`!`	Ciąg jest posortowany poprawnie

Odpowiada ona **pierwszemu testowi przykładowemu**. Liczba wykonanych operacji to 5 (3 porównania i 2 odwrócenia), natomiast suma długości to $4 + 3 = 7$.

Narzędzia do testowania lokalnego i testy przykładowe

W zadaniu są 2 testy przykładowe. Zostanie również udostępniony program **interaktor.cpp** oraz przykładowe błędne rozwiązanie o poprawnym schemacie komunikacji **abc.cpp**. Aby uruchomić rozwiązanie na teście, należy skompilować rozwiązanie i interaktor, a następnie użyć komendę:

`[plik wykonywalny z interaktor.cpp] [test] [plik wykonywalny z rozwiązaniem]`

Na przykład: `./interaktor 0a.in ./abc`