Tempo biegu (A)

Limit pamięci: 32 MB Limit czasu: 0.50 s

Jasio przygotowuje się do biegu na dziesięć kilometrów. Dotychczas trenował na automatycznej bieżni, która pozwalała mu ustawić prędkość w kilometrach na godzinę. Wystarczyło "jedynie" nadążyć za przesuwającym się pasem bieżni. Właściwy bieg odbędzie się jednak na świeżym powietrzu, dlatego Jasio musi nauczyć się kontrolować tempo biegu. W tym celu zakupił zegarek sportowy, który na podstawie pozycji z GPS wyświetla mu jak szybko biegnie. Niestety, zegarek zamiast wyświetlać prędkość w kilometrach na godzinę, wyświetla tempo biegu na jeden kilometr (czyli czas w minutach i sekundach jaki potrzeba na pokonanie dystansu jednego kilometra). Jasio zna swoją idealną prędkość jaką potrafił utrzymać na bieżni. Potrzebuje teraz programu, który przeliczy mu tę prędkość na tempo biegowe zgodnie ze wskazaniem jego zegarka.

Napisz program, który: wczyta idealną prędkość biegu w kilometrach na godzinę, obliczy tempo biegu, które powinien wtedy wyświetlać zegarek i wypisze wynik na standardowe wyjście.

Wejście

W pierwszym (jedynym) wierszu wejścia znajduje się jedna liczba rzeczywista V, podana z dokładnością do jednej cyfry po kropce dziesiętnej. Oznacza ona prędkość biegu w kilometrach na godzinę.

Wyjście

W pierwszym (jedynym) wierszu wyjścia powinno się znaleźć tempo biegu w formacie `MM:SS` (minuty i sekundy, oddzielone dwukropkiem, po dwie cyfry na każdą część, ewentualnie dopełnione zerami wiodącymi). Czas kilometra powinien być zaokrąglony w dół do najbliższej sekundy.

Ograniczenia

$5 \leq V \leq 50$.

Przykład

Wejście	Wyjście	Wyjaśnienie
`10.0`	`06:00`	Prędkość $10\frac{\text{km}}{\text{h}}$ odpowiada przebieganiu jednego kilometra w czasie dokładnie 6 minut.

Wejście	Wyjście	Wyjaśnienie
`11.0`	`05:27`	Dokładny wynik (bez zaokrąglenia) wyniósłby 5 minut $27\frac{3}{11}$ sekund. Należy go wypisać zaokrąglając w dół do najbliższej sekundy.

Wejście	Wyjście
`8.7`	`06:53`

Antypalindromiczność (B)

Limit pamięci: 32 MB Limit czasu: 0.50 s

Jasio obchodzi dzisiaj dzień chłopaka. Tak, wiem, że dzisiaj nie ma dnia chłopaka, ale to jest Jasio, a on jest trochę dziwny i lubi wymyślać. Na przykład wymyślił też, że chce od Ciebie na prezent napis złożony z małych liter alfabetu angielskiego. Niby niewiele, ale wczoraj wymyślił sobie (i dzisiaj jeszcze nie zapomniał), że bardzo nie lubi palindromów, czyli słów, które czytane od lewej do prawej brzmią tak samo jak czytane od prawej do lewej.

Czyli co? Wystarczy dać mu słowo, które nie jest palindromem? Niestety, to byłoby zbyt proste. Jasio ma życzenie, żeby słowo miało N znaków, składało się z jak najmniejszej liczby różnych liter oraz nie może w sobie zawierać (jako spójny fragment) żadnego palindromu długości co najmniej 2. Przykładowo, jeżeli $N = 8$, to słowo `turnieje` odpada, bo zawiera w sobie palindromiczny fragment `eje`. Odpada również dlatego, bo zawiera zbyt wiele różnych liter (istnieją słowa spełniające warunki zadania składające się z mniejszej liczby różnych liter).

Przygotuj dla Jasia prezent i napisz program, który wygeneruje mu napis długości N, składający się z jak najmniejszej liczby różnych liter i nie zawierający żadnego palindromu długości 2 lub więcej jako spójny fragment.

Wejście

W pierwszym (jedynym) wierszu wejścia znajduje się jedna liczba naturalna N, określająca oczekiwaną długość napisu.

Wyjście

W pierwszym (jedynym) wierszu wyjścia powinien się znaleźć ciąg małych liter alfabetu angielskiego – wygenerowany napis dla Jasia.

Jeżeli istnieje wiele możliwych rozwiązań, Twój program może wypisać dowolne z nich.

Ograniczenia

$1 \leq N \leq 500\,000$.

Przykład

Wejście	Wyjście
5	abcab

Maxdiff (c)

Limit pamięci: 256 MB Limit czasu: 2.00 s

Jasio, tak jak wielu innych studentów takich jak on, oblał ostatnio egzamin z Algorytmów i Struktur Danych. Na egzaminie pojawiło się zadanie o przygotowaniu struktury danych, która umożliwia zarządzanie zbiorem liczb naturalnych z operacjami wstawiania elementu, usuwania elementu i obliczania tzw. *mindiff*'a, czyli najmniejszej różnicy między istniejącymi elementami w zbiorze.

Jasio bardzo przejął się swoją porażką i bardzo dobrze przygotował do poprawki. Okazało się, że na egzaminie poprawkowym pojawiło się bliźniaczo podobne zadanie, w którym należy utrzymywać zbiór liczb naturalnych z operacjami:

- `insert` x – wstaw liczbę x do zbioru (lub zignoruj operację, jeżeli liczba x znajduje się już w zbiorze),
- `erase` x – usuń liczbę x ze zbioru (lub zignoruj operację, jeżeli liczba x nie znajduje się w zbiorze),
- `maxdiff` – podaj największą różnicę między elementami w zbiorze (lub zwróć 0, jeżeli moc zbioru jest mniejsza niż 2).

Jasio (może nie bez przyczyny) liczył, że zadania na egzaminie poprawkowym będą identyczne z tymi, które były na pierwszym terminie i niestety nie przygotował się na taką ewentualność i dlatego ponownie nie zaliczył egzaminu. A czy Tobie udałaby się ta sztuka?

Napisz program, który: wczyta operacje do struktury danych opisanej powyżej i wypisze odpowiedzi na wszystkie zapytania `maxdiff`.

Wejście

W pierwszym wierszu wejścia znajduje się jedna liczba naturalna Q, określająca liczbę operacji. W kolejnych Q wierszach znajduje się opis kolejnych operacji, zgodnie z kolejnością ich wykonywania na strukturze danych. Opis każdej operacji składa się z jej nazwy (`insert`, `erase` lub `maxdiff`) oraz, w przypadku `insert` i `erase`, liczby naturalnej x_i (podanej po pojedynczym odstępie w tym samym wierszu).

Wyjście

Twój program powinien wypisać odpowiedzi na wszystkie zapytania `maxdiff` w kolejności ich występowania (i według stanu struktury na moment występowania tych operacji).

Ograniczenia

$1 \leq Q \leq 1\,000\,000$, $1 \leq x_i \leq 10^9$.

Przykład

Wejście	Wyjście
12 maxdiff insert 7 insert 10 insert 9 maxdiff insert 5 maxdiff erase 10 maxdiff erase 10 insert 5 maxdiff	0 3 5 4 4

Dobry LIS (D)

Limit pamięci: 256 MB Limit czasu: 8.00 s

Jasio, przygotowując się do konkursów informatycznych, nauczył się ostatnio rozwiązywać problem LISa (longest incrementing subsequence), czyli zadanie, w którym dany jest ciąg liczb naturalnych i należy z niego wykreślić jak najmniej elementów, aby pozostałe elementy, czytane od lewej do prawej, tworzyły ciąg rosnący (czyli właśnie LIS). Przykładowo: dla ciągu wejściowego $(2, 7, 5, 4, 6, 3, 10, 7, 15)$, LISem jest podciąg $(2, 4, 6, 10, 15)$ (powstały po usunięciu elementów $7, 5, 3, 7$).

Jasio nie poprzestaje, gdy nauczy się czegoś nowego. Wymyślił więc koncept *dobrego LISa*: jest to najdłuższy podciąg rosnący, w którym spełniony jest dodatkowy warunek: suma każdych dwóch jego sąsiednich elementów jest liczbą pierwszą. Dla naszego przykładowego wejściowego ciągu, dobrym LISem jest $(2, 5, 6, 7)$. Sumy sąsiednich elementów są w tym ciągu liczbami pierwszymi ($2 + 5 = 7$, $5 + 6 = 11$, $6 + 7 = 13$). Czy już wiesz jakie będzie zadanie?

Napisz program, który: wczyta ciąg wejściowy, wyznaczy jego dobry LIS i wypisze wynik na standardowe wyjście.

Dla uproszczenia: Możesz założyć, że ciąg wejściowy jest generowany w sposób pseudolosowy: dla każdego testu ustalana jest długość ciągu i wartość M, a potem każdy z N elementów generowanego ciągu jest losowany niezależnie z przedziału $[1, M]$.

Wejście

W pierwszym wierszu wejścia znajduje się jedna liczba naturalna N, określająca długość ciągu wejściowego. W drugim (ostatnim) wierszu wejścia znajduje się ciąg N liczb naturalnych A_i, pooddzielanych pojedynczymi odstępami. Są to kolejne elementy ciągu wejściowego.

Wyjście

W pierwszym wierszu wyjścia powinna się znaleźć jedna liczba naturalna R, określająca długość dobrego LISa. W drugim (ostatnim) wierszu wyjścia powinien się znaleźć rosnący ciąg R liczb naturalnych pooddzielanych pojedynczymi odstępami – dobry LIS ciągu wejściowego.

Jeżeli istnieje wiele możliwych rozwiązań, Twój program może wypisać dowolne z nich.

Ograniczenia

$1 \leq N \leq 200\,000$, $1 \leq A_i \leq 10^9$.

Przykład

Wejście	Wyjście	Wyjaśnienie
9 2 7 5 4 6 3 10 7 15	4 2 5 6 7	Test przykładowy odpowiada przykładowi opisanemu powyżej.

Zliczanie MISów (E)

Limit pamięci: 64 MB Limit czasu: 4.00 s

Jasio, czekając na Twoją pomoc z obliczaniem dobrych LISów (*longest increasing subsequence*), postanowił pomyśleć nad czymś łatwiejszym. Tym razem padło na MISy (*maximal increasing subsequence*).

Jak zapewne wiesz, podciągiem ciągu A nazywamy dowolny ciąg, który można uzyskać, usuwając niektóre (być może żadnego lub nawet wszystkie) elementy A i odczytując pozostawione elementy od lewej do prawej. Podciąg rosnący to podciąg, w którym każdy kolejny element jest większy od poprzedniego. Podciąg rosnący jest maksymalny (jest MISem), gdy przywrócenie ("od-usunięcie") dowolnego elementu z oryginalnego ciągu spowodowałoby, że nie byłby to już podciąg rosnący. Inaczej mówiąc jest to podciąg rosnący, którego nie można już lokalnie powiększyć przywracając do niego żadnego elementu z A.

Zwróć uwagę, że MIS niekoniecznie musi być LISem (najdłuższym podciągiem rosnącym). Przykładowo: dla ciągu $(1, 5, 4, 20, 10, 15)$ zarówno $(1, 5, 10, 15)$ jak i $(1, 5, 20)$ są MISami, choć tylko ten pierwszy jest LISem. Są jeszcze dwa inne MISy – jeden z nich jest trzyelementowy, a jeden czteroelementowy.

Jasio chciałby dla ustalonego ciągu A wyznaczyć liczbę różnych jego MISów. Dwa MISy uznajemy za różne, gdy podzbiory wybranych pozycji w ciągach są różne. Ponieważ Jasio podejrzewa, że ta liczba może być duża i jest to typowe zadanie algorytmiczne, w którym nie chcemy, żebyś implementował(a) arytmetykę dużych liczb, Jasiowi wystarczy poznać resztę z dzielenia tego wyniku przez $10^9 + 7$. Czy pomożesz mu w tym zadaniu?

Napisz program, który: wczyta ciąg A, wyznaczy liczbę jego maksymalnych podciągów rosnących i wypisze wynik na standardowe wyjście.

Dla uproszczenia: Możesz założyć, że ciąg wejściowy jest generowany w sposób pseudolosowy: dla każdego testu ustalana jest długość N ciągu i wartość $M \geq N$, a potem każdy z N elementów generowanego ciągu jest losowany niezależnie z przedziału $[1, M]$.

Wejście

W pierwszym wierszu wejścia znajduje się jedna liczba naturalna N, określająca długość ciągu A. W drugim wierszu wejścia znajduje się ciąg N liczb naturalnych A_i, pooddzielanych pojedynczymi odstępami. Są to kolejne elementy ciągu A.

Wyjście

W pierwszym (jedynym) wierszu wyjścia powinna się znaleźć jedna nieujemna liczba całkowita – reszta z dzielenia przez $10^9 + 7$ liczby MISów w ciągu podanym na wejściu.

Ograniczenia

$1 \leq N \leq 200\,000$, $1 \leq A_i \leq 10^9$.

Przykład

Wejście	Wyjście	Wyjaśnienie
6 1 5 4 20 10 15	4	Test przykładowy odpowiada temu opisanemu powyżej w treści zadania.

Kwadrat z punktami (F)

Limit pamięci: 256 MB Limit czasu: 8.00 s

Jasio, jak to zwykle bywa na tym turnieju, ma problem. Ma zbiór N punktów na płaszczyźnie. Chciałby przykryć co najmniej K spośród nich kwadratem o jak najmniejszym boku. Dla uproszczenia, żeby w zadaniu nie było zbyt dużo prawdziwej geometrii, Jasio akceptuje jedynie kwadraty o bokach równoległych do osi układu współrzędnych. Zakładamy również, że jeżeli kwadrat dotyka jakiegoś punktu, to go przykrywa. Czy pomożesz mu w tym zadaniu?

Napisz program, który: wczyta współrzędne punktów oraz wartość K, wyznaczy optymalny kwadrat przykrywający co najmniej K spośród punktów podanych na wejściu i wypisze wynik na standardowe wyjście.

Wejście

W pierwszym wierszu wejścia znajdują się dwie liczby naturalne N oraz K, oddzielone pojedynczym odstępem. Oznaczają one kolejno: liczbę punktów na płaszczyźnie oraz minimalną liczbę punktów, które należy przykryć.

W kolejnych N wierszach znajduje się opis kolejnych punktów. Opis każdego punktu składa się z dwóch liczb naturalnych x_i oraz y_i, oddzielonych pojedynczym odstępem. Oznaczają one, że i-ty punkt znajduje się na pozycji (x_i, y_i).

Możesz założyć, że współrzędne wszystkich punktów są parami różne.

Wyjście

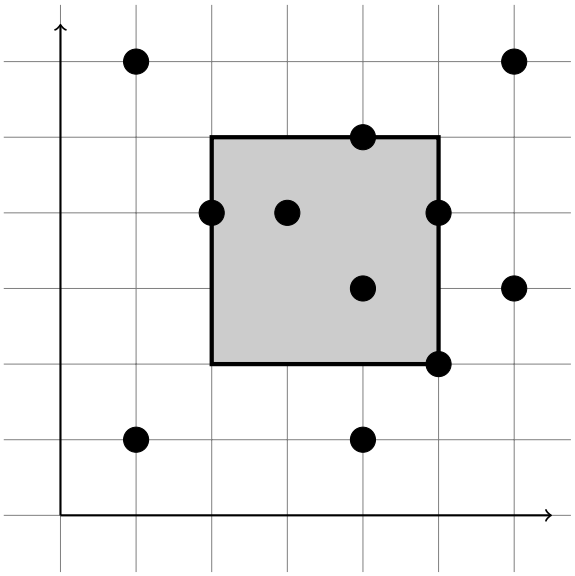
W pierwszym (jedynym) wierszu wyjścia powinna się znaleźć jedna liczba całkowita – minimalna długość boku kwadratu, który (przy swoim optymalnym ułożeniu, zgodnie z warunkami zadania) może przykryć co najmniej K punktów podanych na wejściu.

Ograniczenia

$1 \leq K \leq N \leq 2\,000$, $1 \leq x_i, y_i \leq 10^9$.

Przykład

Wejście	Wyjście
11 6 1 1 3 4 4 3 6 6 1 6 6 3 2 4 4 1 4 5 5 2 5 4	3



Superpermutacja (G)

Limit pamięci: 256 MB Limit czasu: 2.00 s

Jasio, Jasio, Jasio. Możesz już mieć dość zadań o Jasiu, dlatego to zadanie będzie inne i nie będzie już nic o Jasiu. Bo nadszedł czas, żeby w końcu na turnieju rozwiązać jakiś prawdziwy problem matematyczny. Taki, którego jeszcze nikt wcześniej (no, może poza autorem tego zadania, który musiał przecież zaimplementować wzorcówkę) nie rozwiązał. Najlepiej przeczytaj po prostu historię, bo jest "z życia wzięta".

Istnieje taki problem matematyczny (nazwijmy go roboczo problemem *superpermutacji*), w którym chcemy wygenerować możliwie krótki napis, zawierający w sobie (jako spójne fragmenty) wszystkie permutacje napisu $123\ldots N$. Przykładowo: dla $N = 2$ wystarczy wziąć napis `121`, który zawiera w sobie zarówno fragment `12` oraz `21`. Dla $N = 3$, pasuje na przykład `123121321` (który zawiera jako spójne fragmenty wszystkie te napisy: `123`, `132`, `213`, `231`, `312` oraz `321`). Naturalnym jest oczywiście pytanie: ile znaków musi mieć najkrótszy napis zawierający wszystkie permutacje, dla ustalonego N? Problem jest na tyle ciekawy, że pojawiły się nawet interesujące filmy popularnonaukowe na jego temat. Przykładowo: `https://www.youtube.com/watch?v=wJGE4aEWc28`

W dużym skrócie: przez jakiś czas wierzono (postawiono hipotezę), że optymalny napis ma zawsze $1! + 2! + 3! + \ldots + N!$ znaków. Istnieje prosta konstrukcja, która generuje ciąg takiej długości. Została ona opisana na przykład tutaj: `https://njohnston.ca/2013/04/the-minimal-superpermutation-problem/`

Jakiś czas później okazało się jednak, że hipoteza nie jest prawdziwa. Dla $N = 6$ udało się bowiem skonstruować napis o długości 872 (zamiast oczekiwanego 873). Konstrukcja oraz sam napis, podane zostały w tej pracy: `https://arxiv.org/abs/1408.5108`

Autora tej pracy zaproszono nawet do wystąpienia na innym popularnonaukowym kanale, żeby opowiedział o swoim osiągnięciu. Możesz zobaczyć jego opowieść tutaj: `https://www.youtube.com/watch?v=OZzIvl1tbPo`

Jak widać, czasami okazuje się, że gdy hipotezie brakuje dowodu, to może być ona fałszywa. No dobrze, ale gdzie tutaj ten zapowiadany *prawdziwy problem matematyczny*? Czas wcielić się w rolę naukowców przełamujących bariery. Rozważamy problem superpermutacji dla $N = 9$. Celem jest napisanie programu, który wygeneruje napis krótszy o co najmniej 5 znaków od tego, który wynikałby z hipotezy. Należy więc napisać program, który na wyjście wypisuje napis zawierający wszystkie permutacje napisu `123456789` jako spójne podsłowa, którego długość nie przekracza $409\,108$. Powodzenia!

Wejście

W tym zadaniu nie ma wejścia. Twój program nie musi nic wczytywać. Możesz założyć, że jest tylko jeden test, w którym rozpatrujemy zadanie dla $N = 9$.

Wyjście

W pierwszym (jedynym) wierszu wyjścia powinien się znaleźć ciąg znaków `1-9` (bez żadnych odstępów) o długości co najwyżej $1! + 2! + 3! + \ldots + N! - 5$ (dla $N = 9$ suma ta jest równa $409\,108$). Każda spośród wszystkich $N!$ permutacji napisu $123\ldots N$ ma występować choć raz w wygenerowanym napisie jako spójny fragment.

Przykład

W tym zadaniu podawanie przykładu nie ma sensu. Możesz założyć, że gdybyśmy rozpatrywali zadanie dla $N = 6$, a limit długości wynosiłby $1! + 2! + 3! + 4! + 5! + 6! - 1$, to program mógłby po prostu wypisać napis z pracy Houstona.

Funkcja różnowartościowa (H)

Limit pamięci: 512 MB Limit czasu: 5.00 s

Jasio, na przedmiocie *Logika dla informatyków*, dowiedział się ostatnio o istnieniu takiego tworu jak funkcje różnowartościowe. Są to funkcje, które dla dowolnych dwóch różnych argumentów z dziedziny zwracają różne wartości. Przykładowo funkcja $f : \mathbb{R} \to \mathbb{R}$, zadana wzorem $f(x) = 2x + 1$ jest różnowartościowa, ale gdyby była zadana wzorem $f(x) = x^2$ to już nie byłaby różnowartościowa (bo $f(1) = f(-1) = 1$, czyli są dwa różne argumenty dla których wartość funkcji jest taka sama).

Jasio dostał właśnie w prezencie (na dzień chłopaka) funkcję różnowartościową $f : \{1, 2, \ldots, N\} \to \mathbb{N}_+$. Samo to, że funkcja jest różnowartościowa mu się oczywiście bardzo podoba, ale wartości funkcji w niektórych (no dobra, w niektórych testach może być nawet tak, że we wszystkich) punktach go irytują i chciałby je zmienić na inne. Dla każdego argumentu $x \in \{1, 2, \ldots, N\}$ podał swoją idealną wartość $g(x)$, jaką chciałby, żeby przyjęła jego funkcja f. Możliwe, że dla niektórych argumentów x zachodzi $f(x) = g(x)$.

Jasio chciałby wymienić wartości $f(x)$ na $g(x)$ dla jak największej liczby argumentów x, pozostawiając wartości dla pozostałych argumentów niezmienione. Chce przy tym, żeby jego powstała funkcja była różnowartościowa. No a niestety nikt nie powiedział, że Jasio ma trochę oleju w głowie i że g jest różnowartościowa. Oczywiście, jak to zwykle w takich przypadkach bywa, Jasio prosi Cię o pomoc w tym arcyważnym zadaniu.

Napisz program, który: wczyta rozmiar N dziedziny funkcji różnowartościowej f, bieżące wartości tej funkcji $f(x)$ we wszystkich punktach z dziedziny oraz oczekiwane przez Jasia wartości $g(x)$ tej funkcji we wszystkich punktach z dziedziny, wyznaczy optymalny sposób zmiany funkcji na funkcję różnowartościową f' (aby dla jak największej liczby argumentów jej wartość była zgodna z g) według zasad opisanych powyżej i wypisze wynik na standardowe wyjście.

Wejście

W pierwszym wierszu wejścia znajduje się jedna liczba naturalna N określająca rozmiar dziedziny funkcji f. W drugim wierszu wejścia znajduje się ciąg N liczb naturalnych $f(i)$, pooddzielanych pojedynczymi odstępami. Są to wartości funkcji f w kolejnych punktach $1, 2, \ldots, N$. W trzecim (ostatnim) wierszu wejścia znajduje się ciąg N liczb naturalnych $g(i)$ pooddzielanych pojedynczymi odstępami. Są to oczekiwane wartości g w kolejnych punktach $1, 2, \ldots, N$.

Wyjście

W pierwszym wierszu wyjścia powinna się znaleźć jedna nieujemna liczba całkowita – największa liczba argumentów x, dla których $f'(x) = g(x)$.

W drugim (ostatnim) wierszu wyjścia powinien się znaleźć ciąg N liczb naturalnych pooddzielanych pojedynczymi odstępami – wartości $f'(1), f'(2), \ldots, f'(N)$.

Jeżeli istnieje wiele rozwiązań, Twój program może wypisać dowolne z nich.

Ograniczenia

$1 \leq N \leq 500\,000$, $1 \leq f(x), g(x) \leq 10^9$.

Przykład

Wejście	Wyjście
9 5 8 3 10 7 20 4 6 12 8 5 7 7 3 20 6 9 6	7 8 5 7 10 3 20 6 9 12

Zwiększanie wyniku (I)

Limit pamięci: 32 MB Limit czasu: 0.25 s

Jak już wiesz z innych zadań, Jasio startuje w zawodach informatycznych i nie idzie mu zbyt dobrze, bo ciągle prosi Cię o pomoc. W ostatnim konteście udało mu się (poprosić kogoś, żeby) złamać zabezpieczenia sprawdzaczki i dzięki temu może sobie powiększyć wynik niektórych zadań. Dokładniej: Jasio może K-krotnie zwiększyć wynik z jakiegoś zadania o 1 punkt. Każde zwiększenie działa w sposób niezależny: Jasio może wszystkie zwiększenia użyć na jednym zadaniu lub jakoś inteligentnie je rozdzielać między zadaniami. Punktacja kontestu jest jednak nieco inna niż zwykle, bo wynik zawodnika to nie suma, a **iloczyn** wyników poszczególnych zadań. Należałoby teraz jedynie wybrać, gdzie należy zużyć swoje zhackowane zwiększenia, żeby zmaksymalizować ów finalny wynik. Jak się zapewne domyślasz, Jasio będzie miał z tym problem i oczywiście przyszedł po pomoc do Ciebie. Powodzenia!

Napisz program, który wczyta wyniki Jasia oraz wartość K, obliczy maksymalny możliwy do uzyskania finalny wynik Jasia na konteście po zwiększeniach i wypisze wynik na standardowe wyjście.

Wejście

W pierwszym wierszu wejścia znajdują się dwie liczby naturalne N oraz K, oddzielone pojedynczym odstępem. Są to odpowiednio: liczba zadań na konteście oraz maksymalna liczba zwiększeń wyniku zadania (o 1) dostępna dla Jasia. W drugim (ostatnim) wierszu wejścia znajduje się ciąg N nieujemnych liczb całkowitych pooddzielanych pojedynczymi odstępami. Są to wyniki Jasia na poszczególnych zadaniach.

Wyjście

W pierwszym (jedynym) wierszu wyjścia powinna się znaleźć jedna nieujemna liczba całkowita – maksymalny możliwy do osiągnięcia iloczyn wyników zadań po wykonaniu zwiększeń.

Uwaga: Możesz założyć, że dane są tak dobrane, że ów wynik nie przekracza 10^{18}.

Ograniczenia

$1 \leq N \leq 500\,000$, $1 \leq K \leq 10^9$, $0 \leq A_i \leq 10^9$.

Przykład

Wejście	Wyjście
3 2 2 2 2	18

Wejście	Wyjście
5 4 0 0 0 0 0	0

Wejście	Wyjście
3 2 0 0 100	100

Wejście	Wyjście
4 3 0 100 100 100	3000000