Problem A. Palindromes

Input file:	`palindrome.in`
Output file:	`palindrome.out`
Time limit:	1 second
Memory limit:	128 megabytes

You are given a string of lower-case Latin letters. Let us define substring's "occurrence value" as the number of the substring occurrences in the string multiplied by the length of the substring. For a given string find the largest occurrence value of palindromic substrings.

Input

The only line of input contains a non-empty string of lower-case Latin letters (a-z).

Output

Output one integer – the largest occurrence value of palindromic substrings.

Examples

palindrome.in	palindrome.out
abacaba	7
www	4

Note

$|s|$ is length of string s.

A substring of string $s_1 s_2 \ldots s_{|s|}$ is any non-empty string $s_i s_{i+1} \ldots s_j$, where $1 \le i \le j \le |s|$. Any string is also its own substring.

A string is called palindromic, if it is read the same in either direction, from left to right and from right to left.

In the first sample there are seven palindromic substrings $a, b, c, aba, aca, bacab, abacaba$.

- a has 4 occurrences in the given string, its occurrence value is $4 \times 1 = 4$
- b has 2 occurrences in the given string, its occurrence value is $2 \times 1 = 2$
- c has 1 occurrence in the given string, its occurrence value is $1 \times 1 = 1$
- aba has 2 occurrences in the given string, its occurrence value is $2 \times 3 = 6$
- aca has 1 occurrence in the given string, its occurrence value is $1 \times 3 = 3$
- $bacab$ has 1 occurrence in the given string, its occurrence value is $1 \times 5 = 5$
- $abacaba$ has 1 occurrence in the given string, its occurrence value is $1 \times 7 = 7$

So, the largest occurrence value of palindromic substrings is 7.

Scoring

Your program will be tested on 5 sets of input instances as follows:

Subtask 1 (points: 8)

$1 \le |s| \le 100$.

Subtask 2 (points: 15)

$1 \leq |s| \leq 1000$.

Subtask 3 (points: 24)

$1 \leq |s| \leq 10000$.

Subtask 4 (points: 26)

$1 \leq |s| \leq 100000$.

Subtask 5 (points: 27)

$1 \leq |s| \leq 300000$.

Problem B. Split the sequence

Input file:	`sequence.in`
Output file:	`sequence.out`
Time limit:	2 seconds
Memory limit:	128 megabytes

You are playing a game with a sequence of n non-negative integers. In this game you have to split the sequence into $k+1$ non-empty parts. To obtain $k+1$ parts you repeat the following steps k times:

1. Choose any part with more than one element (initially you have only one part – the whole sequence).
2. Split it between any two elements to get two new non-empty parts.

Each time after these steps you gain the number of points which is equal to product of sums of elements of each new part. You want to maximize the total number of points you gain.

Input

The first line of the input file contains two integers n and k ($k+1 \le n$). The second line of input contains n non-negative integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^4$) – the sequence.

Output

On the first line output the largest total number of points you can gain. On the second line output k integers between 1 and $n-1$ — the positions of elements after which you have to split the sequence to gain the largest total number of points. If there are more than one way to gain the largest number of points output any one of them.

Examples

sequence.in	sequence.out
7 3 4 1 3 4 0 2 3	108 1 3 5

Note

In the first sample you can gain 108 points by the following way:

- Initially you have the whole sequence (4, 1, 3, 4, 0, 2, 3) as one part. You will split the sequence after 1st element and gain $4 \times (1+3+4+0+2+3) = 52$ points.
- You have two parts (4), (1, 3, 4, 0, 2, 3). You will split the sequence after 3rd element and gain $(1+3) \times (4+0+2+3) = 36$ points.
- You have three parts (4), (1,3), (4, 0, 2, 3). You will split the sequence after 5th element and gain $(4+0) \times (2+3) = 20$ points.

So, after the steps taken above you get four parts (4), (1,3), (4,0), (2, 3) and gain $52+36+20 = 108$ points.

Scoring

Your program will be tested on 6 sets of input instances as follows:

Subtask 1 (points: 11)

$1 \le k < n \le 10$.

Subtask 2 (points: 11)

$1 \leq k < n \leq 50$.

Subtask 3 (points: 11)

$1 \leq k < n \leq 200$.

Subtask 4 (points: 17)

$2 \leq n \leq 1000$, $1 \leq k \leq min(n-1, 200)$.

Subtask 5 (points: 21)

$2 \leq n \leq 10000$, $1 \leq k \leq min(n-1, 200)$.

Subtask 6 (points: 29)

$2 \leq n \leq 100000$, $1 \leq k \leq min(n-1, 200)$.

Problem C. Beads and wires

Input file:	`beads.in`
Output file:	`beads.out`
Time limit:	1 second
Memory limit:	128 megabytes

A popular child game at the time of Leonardo was called Beads-and-wires. Not surprisingly, it was played with beads and wires. The wires are red or blue. The beads are numbered from 1 to n. The game starts with a single bead. From this moment on, new beads can be added using wires as follows.

- Append(w, v): a new bead w is attached to an existing bead v through a piece of red wire.
- Insert(w, u, v): a new bead w is inserted by substituting an already existing red wire connecting existing beads u and v, with two new blue wires by putting w between u and v; in other words, the existing red wire $u - v$ is replaced by two new blue wires $u - w$ and $w - v$.

Every piece of wire (both blue or red) has a certain length. At the end of the game, what counts is the sum of lengths of blue wires only (the length of red wires doesn't count): this is called the final score.

You are given the final configuration reached by a beads-and-wire game, specifying how beads are connected to one another and providing also the length of each piece of wire, but omitting the color of the wires.

You have to write a program that finds the maximal possible final score for that configuration. More precisely, among all the feasible beads-and-wire games that end with that configuration, you have to find one that has the maximum final score (sum of blue wire length) and you have to output that value.

Input

First line contains positive integer n — number of beads, which are numbered from 1 to n. Next $n - 1$ lines contains 3 numbers each: a_i, b_i ($1 \leq a_i < b_i \leq n$) and $1 \leq c_i \leq 10000$ — two beads connected by a piece of wire, and the length of that piece.

Output

Output one integer – the maximum final score.

Examples

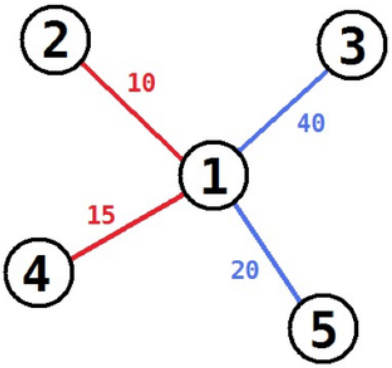
beads.in	beads.out
5 1 2 10 1 3 40 1 4 15 1 5 20	60
10 4 10 2 1 2 21 1 3 13 6 7 1 7 9 5 2 4 3 2 5 8 1 6 55 6 8 34	140

Note

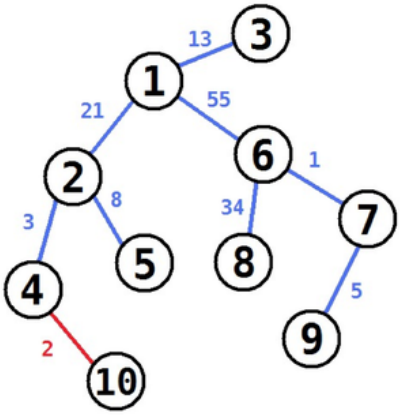
For the first sample testcase, the final score of 60 can be obtained as follows: start with the single bead 3.

- append 5 to 3 (with a wire of arbitrary length).
- insert 1 on the wire $3 - 5$ (using wires of length 40 and 20).
- append 2 to 1 with a wire of length 10.
- append 4 to 1 with a wire of length 15.

The configuration obtained this way is shown in the first figure. It is possible to see that there is no way to obtain the same configuration (apart for the colors) with a larger final score.



For second sample testcase the configuration obtained this way is shown in the second figure, and has a final score of 140.



Scoring

Your program will be tested on 4 sets of input instances as follows:

Subtask 1 (points: 13)

$1 \leq n \leq 10$.

Subtask 2 (points: 15)

$1 \leq n \leq 200$.

Subtask 3 (points: 29)

$1 \leq n \leq 10000$.

Subtask 4 (points: 43)

$1 \leq n \leq 200000$.