# Festival

Nayra is at a festival playing a game where the grand prize is a trip to Laguna Colorada. The game consists of using tokens to buy coupons. Buying a coupon may result in additional tokens. The goal is to get as many coupons as possible.

She starts the game with $A$ tokens. There are $N$ coupons, numbered from $0$ to $N-1$. Nayra has to pay $P[i]$ tokens ($0 \le i < N$) to buy coupon $i$ (and she must have at least $P[i]$ tokens before the purchase). She can buy each coupon at most once.

Moreover, each coupon $i$ ($0 \le i < N$) is assigned a **type**, denoted by $T[i]$, which is an integer **between** $1$ **and** $4$**, inclusive**. After Nayra buys coupon $i$, the remaining number of tokens she has gets multiplied by $T[i]$. Formally, if she has $X$ tokens at some point of the game, and buys coupon $i$ (which requires $X \ge P[i]$), then she will have $(X - P[i]) \cdot T[i]$ tokens after the purchase.

Your task is to determine which coupons Nayra should buy and in what order, to maximize the total number of **coupons** she has at the end. If there is more than one sequence of purchases that achieves this, you may report any one of them.

## Implementation Details

You should implement the following procedure.

```
std::vector<int> max_coupons(int A, std::vector<int> P,
    std::vector<int> T)
```

- $A$: the initial number of tokens Nayra has.
- $P$: an array of length $N$ specifying the prices of the coupons.
- $T$: an array of length $N$ specifying the types of the coupons.
- This procedure is called exactly once for each test case.

The procedure should return an array $R$, which specifies Nayra's purchases as follows:

- The length of $R$ should be the maximum number of coupons she can buy.
- The elements of the array are the indices of the coupons she should buy, in chronological order. That is, she buys coupon $R[0]$ first, coupon $R[1]$ second, and so on.
- All elements of $R$ must be unique.

If no coupons can be bought, $R$ should be an empty array.

## Constraints

- $1 \le N \le 200\,000$
- $1 \le A \le 10^9$
- $1 \le P[i] \le 10^9$ for each $i$ such that $0 \le i < N$.
- $1 \le T[i] \le 4$ for each $i$ such that $0 \le i < N$.

## Subtasks

| Subtask | Score | Additional Constraints |
|---|---|---|
| 1 | $5$ | $T[i] = 1$ for each $i$ such that $0 \le i < N$. |
| 2 | $7$ | $N \le 3000$; $T[i] \le 2$ for each $i$ such that $0 \le i < N$. |
| 3 | $12$ | $T[i] \le 2$ for each $i$ such that $0 \le i < N$. |
| 4 | $15$ | $N \le 70$ |
| 5 | $27$ | Nayra can buy all $N$ coupons (in some order). |
| 6 | $16$ | $(A - P[i]) \cdot T[i] < A$ for each $i$ such that $0 \le i < N$. |
| 7 | $18$ | No additional constraints. |

## Examples

### Example 1

Consider the following call.

```
max_coupons(13, [4, 500, 8, 14], [1, 3, 3, 4])
```

Nayra initially has $A = 13$ tokens. She can buy $3$ coupons in the order shown below:

| Coupon bought | Coupon price | Coupon type | Number of tokens after the purchase |
|---|---|---|---|
| $2$ | $8$ | $3$ | $(13 - 8) \cdot 3 = 15$ |
| $3$ | $14$ | $4$ | $(15 - 14) \cdot 4 = 4$ |
| $0$ | $4$ | $1$ | $(4 - 4) \cdot 1 = 0$ |

In this example, it is not possible for Nayra to buy more than $3$ coupons, and the sequence of purchases described above is the only way in which she can buy $3$ of them. Hence, the procedure should return $[2, 3, 0]$.

### Example 2

Consider the following call.

```
max_coupons(9, [6, 5], [2, 3])
```

In this example, Nayra can buy both coupons in any order. Hence, the procedure should return either $[0, 1]$ or $[1, 0]$.

### Example 3

Consider the following call.

```
max_coupons(1, [2, 5, 7], [4, 3, 1])
```

In this example, Nayra has one token, which is not enough to buy any coupons. Hence, the procedure should return $[\,]$ (an empty array).

## Sample Grader

Input format:

```
N A
P[0] T[0]
P[1] T[1]
...
P[N-1] T[N-1]
```

Output format:

```
S
R[0] R[1] ... R[S-1]
```

Here, $S$ is the length of the array $R$ returned by `max_coupons`.