Task 1. Hint

Doc Brown managed to turn his DeLorean into a time machine. He has now set his sights on an even bigger problem: longest common subsequence. When given two sequences of numbers A and B with lengths N and M, he wants to find the longest (or one of the longest) sequence C, such that all elements of C appear in both A and B in the same order (but not necessarily consecutively) as in C. He has managed to write a somewhat slow program which will run for many days until it finds a solution. However, he needs an answer as soon as possible. His initial plan was to leave his program running and then in a few days send its output back in time to his present self. The problem is that time travel requires tremendous amounts of energy, so sending the full solution back in time would be extremely expensive.

Now Doc has a new plan, but he needs your help to implement it. He wants to send a short hint about the solution from the future to the present and then use this hint to reconstruct an optimal solution using the hint. Note that it does not necessarily need to be the same optimal solution as the one from the future.

You should write a program `hint.cpp` which implements two functions: `genHint` and `solve`, which achieve Doc's plan.

Implementation details

Your function `genHint` needs to have the following prototype:

```
std::vector<bool> genHint(const std::vector<int>& a, const std::vector<int>& b,
    const std::vector<int>& sol);
```

It will get called only once and receive as arguments the two given sequences and an optimal solution. It should return a hint, which will be sent to your other function.

Your function `solve` needs to have the following prototype:

```
std::vector<int> solve(const std::vector<int>& a, const std::vector<int>& b,
    const std::vector<bool>& hint);
```

It will get called only once and receive as arguments the two given sequences and the hint that your other function returned. It should return an optimal solution.

Besides these two functions, your program may have any sort of auxiliary functions, classes, variables, etc. However, it **must not** contain a `main` function and it **must** include the header file `hint.h`. **Note that on the grading system your two functions will be called in separate instances of your program, running in separate processes**. This means you will not be able to share any global state between the runs of the two functions.

Local testing

You are given the file `Lgrader.cpp`, which you can compile together with your program to test it. It will read N and M, followed by A and B. After that it will read the optimal solution length K and an optimal solution C. It will output the length of the hint from your `hint` function, as well as the solution that your `solve` function returned. Note that, unlike in the official grading system, the local grader does not run your functions in separate processes.

Constraints

$1 \leq N, M \leq 2 \times 10^5$
$0 \leq A_i, B_j < \min(N, M)$

Subtasks

Subtask	Points	N, M
1	10	$\leq 10^4$
2	90	$\leq 2 \times 10^5$

Your solution will receive points for a subtask only if it passes all tests in it.

Scoring

The score you receive for a given subtask depends on the maximum length L of a hint your program has generated for a test in that subtask. The part of the points that you receive for the subtask will be:

$$\min\left(\left(\frac{640}{L+1}\right)^{0.3}, 1\right)$$