Exam

To achieve a full score one could solve subtasks 2, 4 and 6 separately.

Subtask 2:

Go over the consecutive intervals of A with works that aren't better than B_1 and accumulate the lengths of those intervals that contain at least one work equal to B_1. Since only one pass is needed it works in O(N).

To solve other two subtasks we will need the following observations:

- The works in the final state will appear in the same relative order as they were originally
- student X may end up with student's Y's work if and only if in the interval between X and Y (inclusive) there were no works that were better than X's work

Any state that satisfies above requirements is reachable.
For each student X we can find the interval containing X and extending on both sides as much as possible that doesn't contain works better than X's. This can be done in multiple ways:

- Simple seek in each direction until we find better work, in $O(N^2)$ which is good enough for most subtasks
- Using data structures (RMQ/Segment tree), in O(N logN)
- Single pass in each direction maintaining sorted stack, in O(N), similar to the one described in the task Fountain

Using these precomputed intervals we can answer whether student Y can end up with student X's work in O(1) and doing these checks at appropriate times will be needed in both subtasks.

Subtask 4:

We have to find the longest subsequence in B such that corresponding values in A are in non-decreasing order. This can be done by solving a well known Longest Increasing Subsequence (LIS) problem with O(N logN) solution.

Subtask 6:

We have to match the most values from B to A in the same order, which can be done in $O(N^2)$ using the standard 2D dynamic programming similar to the one that solves Longest Common Subsequence (LCS) problem.