Climbing the Leaderboard

```
#include <bits/stdc++.h>

using namespace std;

string ltrim(const string &);
string rtrim(const string &);
vector<string> split(const string &);

/*
 * Complete the 'climbingLeaderboard' function below.
 *
 * The function is expected to return an INTEGER_ARRAY.
 * The function accepts following parameters:
 *  1. INTEGER_ARRAY ranked
 *  2. INTEGER_ARRAY player
 */

vector<int> climbingLeaderboard(vector<int> ranked, vector<int>
player) {
    vector<int> scores;
    scores.push_back(ranked[0]);
    for(int i=1;i<ranked.size();i++){
        if(ranked[i]!=ranked[i-1]){
            scores.push_back(ranked[i]);
        }
    }
    vector<int> result;
    int n=scores.size();
    int idx=n-1;
    for(int i=0;i<player.size();i++){
        int score=player[i];
        while(idx>=0 && score>=scores[idx]){
            idx--;
        }
        result.push_back(idx+2);
    }
    return result;
}

int main()
{
    ofstream fout(getenv("OUTPUT_PATH"));
```

```
string ranked_count_temp;
getline(cin, ranked_count_temp);

int ranked_count = stoi(ltrim(rtrim(ranked_count_temp)));

string ranked_temp_temp;
getline(cin, ranked_temp_temp);

vector<string> ranked_temp = split(rtrim(ranked_temp_temp));

vector<int> ranked(ranked_count);

for (int i = 0; i < ranked_count; i++) {
    int ranked_item = stoi(ranked_temp[i]);

    ranked[i] = ranked_item;
}

string player_count_temp;
getline(cin, player_count_temp);

int player_count = stoi(ltrim(rtrim(player_count_temp)));

string player_temp_temp;
getline(cin, player_temp_temp);

vector<string> player_temp = split(rtrim(player_temp_temp));

vector<int> player(player_count);

for (int i = 0; i < player_count; i++) {
    int player_item = stoi(player_temp[i]);

    player[i] = player_item;
}

vector<int> result = climbingLeaderboard(ranked, player);

for (size_t i = 0; i < result.size(); i++) {
    fout << result[i];

    if (i != result.size() - 1) {
        fout << "\n";
    }
}
```

```
    fout << "\n";

    fout.close();

    return 0;
}

string ltrim(const string &str) {
    string s(str);

    s.erase(
        s.begin(),
        find_if(s.begin(), s.end(), not1(ptr_fun<int,
int>(isspace)))
    );

    return s;
}

string rtrim(const string &str) {
    string s(str);

    s.erase(
        find_if(s.rbegin(), s.rend(), not1(ptr_fun<int,
int>(isspace))).base(),
        s.end()
    );

    return s;
}

vector<string> split(const string &str) {
    vector<string> tokens;

    string::size_type start = 0;
    string::size type end = 0;

    while ((end = str.find(" ", start)) != string::npos) {
        tokens.push back(str.substr(start, end - start));

        start = end + 1;
    }

    tokens.push_back(str.substr(start));

    return tokens;
```

}