```cpp
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

#include <algorithm>
#include <vector>
#include <numeric>
using namespace std;

vector<int> getLocalMinIdxs(vector<int> array);
void expandFromLocalMinIdx(int localMinIdx, vector<int> scores,
                           vector<int> *rewards);

// O(n) time | O(n) space - where in is the length of the input array
int minRewards(vector<int> scores) {
  vector<int> rewards = vector<int>(scores.size(), 1);
  vector<int> localMinIdxs = getLocalMinIdxs(scores);
  for (int localMinIdx : localMinIdxs) {
    expandFromLocalMinIdx(localMinIdx, scores, &rewards);
  }
  return accumulate(rewards.begin(), rewards.end(), 0);
}

vector<int> getLocalMinIdxs(vector<int> array) {
  if (array.size() == 1)
    return vector<int>{0};
  vector<int> localMinIdxs = {};
  for (int i = 0; i < array.size(); i++) {
    if (i == 0 && array[i] < array[i + 1])
      localMinIdxs.push_back(i);
    if (i == array.size() - 1 && array[i] < array[i - 1])
      localMinIdxs.push_back(i);
    if (i == 0 || i == array.size() - 1)
      continue;
    if (array[i] < array[i + 1] && array[i] < array[i - 1])
      localMinIdxs.push_back(i);
  }
  return localMinIdxs;
}

void expandFromLocalMinIdx(int localMinIdx, vector<int> scores,
                           vector<int> *rewards) {
  int leftIdx = localMinIdx - 1;
  while (leftIdx >= 0 && scores[leftIdx] > scores[leftIdx + 1]) {
    rewards->at(leftIdx) =
        max(rewards->at(leftIdx), rewards->at(leftIdx + 1) + 1);
    leftIdx--;
  }
  int rightIdx = localMinIdx + 1;
  while (rightIdx < scores.size() && scores[rightIdx] > scores[rightIdx - 1]) {
    rewards->at(rightIdx) = rewards->at(rightIdx - 1) + 1;
    rightIdx++;
  }
}
```