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

// O(n) time | O(n) space - where in is the length of the input array
function minRewards(scores) {
  const rewards = scores.map(_ => 1);
  const localMinIdxs = getLocalMinIdxs(scores);
  for (const localMinIdx of localMinIdxs) {
    expandFromLocalMinIdx(localMinIdx, scores, rewards);
  }
  return rewards.reduce((a, b) => a + b);
}

function getLocalMinIdxs(array) {
  if (array.length === 1) return [0];
  const localMinIdxs = [];
  for (let i = 0; i < array.length; i++) {
    if (i === 0 && array[i] < array[i + 1]) localMinIdxs.push(i);
    if (i === array.length - 1 && array[i] < array[i - 1]) localMinIdxs.push(i);
    if (i === 0 || i === array.length - 1) continue;
    if (array[i] < array[i + 1] && array[i] < array[i - 1]) localMinIdxs.push(i);
  }
  return localMinIdxs;
}

function expandFromLocalMinIdx(localMinIdx, scores, rewards) {
  let leftIdx = localMinIdx - 1;
  while (leftIdx >= 0 && scores[leftIdx] > scores[leftIdx + 1]) {
    rewards[leftIdx] = Math.max(rewards[leftIdx], rewards[leftIdx + 1] + 1);
    leftIdx--;
  }
  let rightIdx = localMinIdx + 1;
  while (rightIdx < scores.length && scores[rightIdx] > scores[rightIdx - 1]) {
    rewards[rightIdx] = rewards[rightIdx - 1] + 1;
    rightIdx++;
  }
}

exports.minRewards = minRewards;

```