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

import java.util.*;
import java.util.stream.*;

// O(n) time | O(n) space - where in is the length of the input array
class Program {
  public static int minRewards(int[] scores) {
    int[] rewards = new int[scores.length];
    Arrays.fill(rewards, 1);
    List<Integer> localMinIdxs = getLocalMinIdxs(scores);
    for (Integer localMinIdx : localMinIdxs) {
      expandFromLocalMinIdx(localMinIdx, scores, rewards);
    }
    return IntStream.of(rewards).sum();
  }

  public static List<Integer> getLocalMinIdxs(int[] array) {
    List<Integer> localMinIdxs = new ArrayList<Integer>();
    if (array.length == 1) {
      localMinIdxs.add(0);
      return localMinIdxs;
    }
    for (int i = 0; i < array.length; i++) {
      if (i == 0 && array[i] < array[i + 1]) localMinIdxs.add(i);
      if (i == array.length - 1 && array[i] < array[i - 1]) localMinIdxs.add(i);
      if (i == 0 || i == array.length - 1) continue;
      if (array[i] < array[i + 1] && array[i] < array[i - 1]) localMinIdxs.add(i);
    }
    return localMinIdxs;
  }

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

```