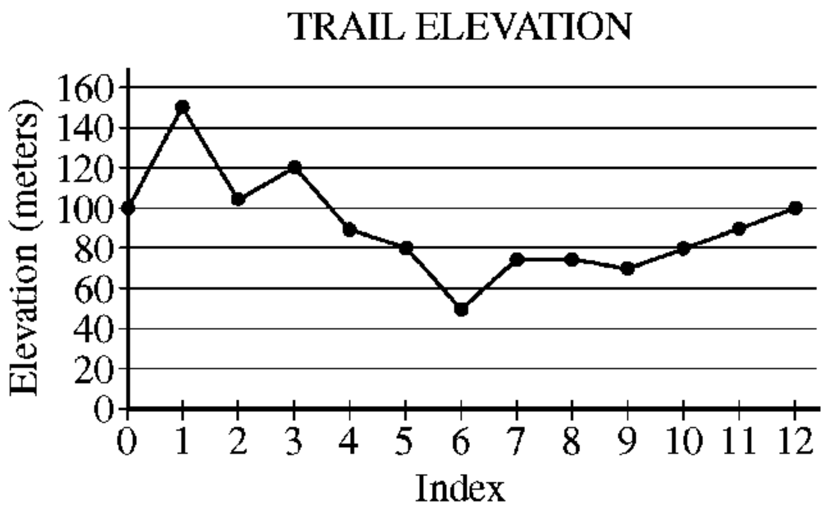
hikingTrailArray

This was an actual AP Free Response question - see if you can write it perfectly the first time with no errors.

A hiking trail has elevation markers posted at regular intervals along the trail. Elevation information about a trail can be stored in an array, where each element in the array represents the elevation at a marker. The elevation at the first marker will be stored at array index 0; the elevation at the second marker will be stored at array index 1, and so forth. Elevations between markers are ignored in this question. The graph below shows an example of trail elevations.



The table below contains the data represented in the graph.

Trail Elevation (meters)

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
Elevation	100	150	105	120	90	80	50	75	75	70	80	90	100

The declaration of the Trail class is shown below. **You will write two unrelated methods of the Trail class.**

```
/*** <<< This code is NOT complete >>>
 * Complete the two methods: isLevelTrailSegment, and isDifficult
 */
public class Trail
{
  /** Representation of the trail. The number of markers on the trail is markers.length. */
  private int[] markers;

  /** Constructs a new Trail object. */
  public Trail(int[] m)
  {
     markers  = new int[m.length];
     for (int i = 0; i < m.length; i++)
       markers[i] = m[i];
  }


  /** Determines if a trail segment is level. A trail segment is defined by a starting
   * marker, an ending marker, and all markers between those two markers.
   * A trail segment is level if it has a difference between the maximum elevation
   * and minimum elevation that is less than or equal to 10 meters.
   * @param start the index of the starting marker
   * @param end the index of the ending marker
   *
   * Precondition: 0 <= start < end <= markers.length - 1
   * @return true if the difference between the maximum and minimum
   * elevation on this segment of the trail is less than or equal to 10 meters;
   * false otherwise.
   */
   public boolean isLevelTrailSegment(int start, int end)
   {
      /*** to be implemented in part (a) ***/
    }

   /** Determines if this trail is rated difficult. A trail is rated by counting the
    * number of changes in elevation that are at least 30 meters (up or down) between
    * two consecutive markers. A trail with 3 or more such changes is rated difficult.
    * @return true if the trail is rated difficult; false otherwise.
    */
```

```
     public boolean isDifficult()
    {
       /*** to be implemented in part (b) */
    }


    /*** The main method is complete ***/
    public static void main(String[] args)
    {
        Trail t1 = new Trail(new int[] {100, 150, 105, 120, 90, 80, 50, 75, 75, 70, 80, 90, 
100});
        System.out.println("Trail t1: {100, 150, 105, 120, 90, 80, 50, 75, 75, 70, 80, 90, 
100}");
        System.out.println("isLevel between 7 and 10 should be <<true>>, your output is 
<<" + 
                          t1.isLevelTrailSegment(7, 10)+">>");
        System.out.println("isLevel between 2 and 12 should be <<false>>, your output is 
<<" + 
                          t1.isLevelTrailSegment(2, 12)+">>");
        System.out.println("isDifficult should be <<true>>, your output is <<" + 
                          t1.isDifficult()+">>");

        Trail t2 = new Trail(new int[] {150, 120, 100, 80, 75, 75, 90, 100});
        System.out.println("\nTrail t2: {150, 120, 100, 80, 75, 75, 90, 100}");
        System.out.println("isDifficult should be <<false>>, your output is <<" + 
                          t2.isDifficult()+">>");

        Trail t3 = new Trail(new int[] {150, 120, 100, 80, 55, 90, 60});
        System.out.println("\nTrail t3: {150, 120, 100, 80, 55, 90, 60}");
        System.out.println("isDifficult should be <<true>>, your output is <<" + 
                          t3.isDifficult()+">>");

    }
}
```