Trail

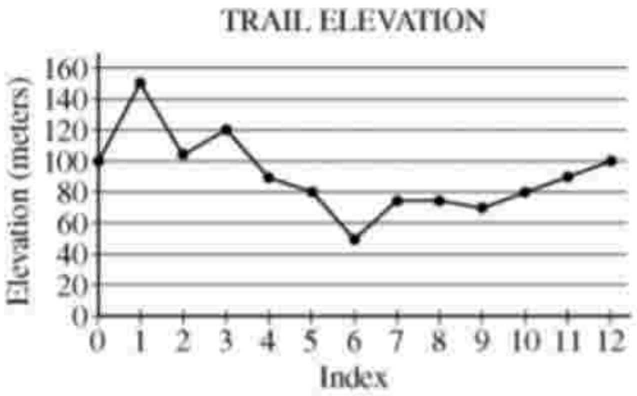
Name

1. **Directions**: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Quick Reference found in the Appendix have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods may not receive full credit.

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

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.

```
public class Trail
{
  /** Representation of the trail. The number of markers on the trail is markers.length. */
  private int[] markers;

  /** 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) */  }

  // There may be instance variables, constructors, and methods that are not shown.
}
```

a. Write the Trail method isLevelTrailSegment. A trail segment is defined by a starting marker, an ending marker, and all markers between those two markers. The parameters of the method are the index of the starting marker and the index of the ending marker. The method will return true if the difference between the maximum elevation and the minimum elevation in the trail segment is less than or equal to 10 meters.

For the trail shown at the beginning of the question, the trail segment starting at marker 7 and ending at marker 10 has elevations ranging between 70 and 80 meters. Because the difference between 80 and 70 is equal to 10, the trail segment is considered level.

The trail segment starting at marker 2 and ending at marker 12 has elevations ranging between 50 and 120 meters. Because the difference between 120 and 50 is greater than 10, this trail segment is not considered level.

Trail

Complete method isLevelTrailSegment below.

```
/** 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)
```

b. Write the Trail method isDifficult. 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. The following table shows trail elevation data and the elevation changes between consecutive trail markers.

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

Elevation change	50	-45	15	-30	-10	-30	25	0	-5	10	10	10

This trail is rated difficult because it has 4 changes in elevation that are 30 meters or more (between markers 0 and 1, between markers 1 and 2, between markers 3 and 4, and between markers 5 and 6).

Complete method isDifficult below.

```
/** Determines if this trail is 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()
```

Please respond on separate paper, following directions from your teacher.