2. An `APLine` is a line defined by the equation $ax + by + c = 0$, where a is not equal to zero, b is not equal to zero, and a, b, and c are all integers. The slope of an `APLine` is defined to be the `double` value $-a/b$. A point (represented by integers x and y) is on an `APLine` if the equation of the `APLine` is satisfied when those x and y values are substituted into the equation. That is, a point represented by x and y is on the line if $ax + by + c$ is equal to 0. Examples of two `APLine` equations are shown in the following table.

Equation	**Slope (–*a* / *b*)**	**Is point (5, -2) on the line?**
$5x + 4y - 17 = 0$	-5 / 4 = -1.25	Yes, because 5(5) + 4(-2) + (-17) = 0
$-25x + 40y + 30 = 0$	25 / 40 = 0.625	No, because -25(5) + 40(-2) + 30 $\neq$ 0

Assume that the following code segment appears in a class other than `APLine`. The code segment shows an example of using the `APLine` class to represent the two equations shown in the table.

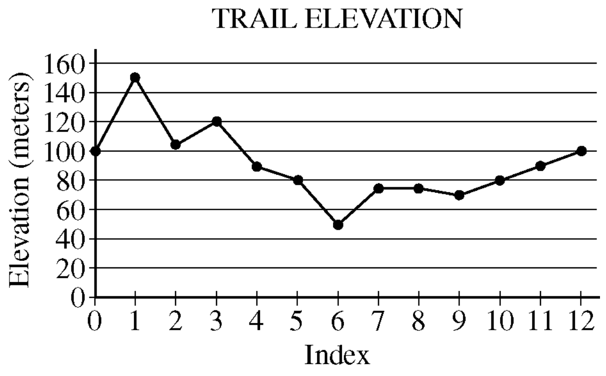
```
APLine line1 = new APLine(5, 4, -17);
double slope1 = line1.getSlope();        // slope1 is assigned -1.25
boolean onLine1 = line1.isOnLine(5, -2); // true because 5(5) + 4(-2) + (-17) = 0


APLine line2 = new APLine(-25, 40, 30);
double slope2 = line2.getSlope();        // slope2 is assigned 0.625
boolean onLine2 = line2.isOnLine(5, -2); // false because -25(5) + 40(-2) + 30 ≠ 0
```

Write the `APLine` class. Your implementation must include a constructor that has three integer parameters that represent a, b, and c, in that order. You may assume that the values of the parameters representing a and b are not zero. It must also include a method `getSlope` that calculates and returns the slope of the line, and a method `isOnLine` that returns `true` if the point represented by its two parameters (`x` and `y`, in that order) is on the `APLine` and returns `false` otherwise. Your class must produce the indicated results when invoked by the code segment given above. You may ignore any issues related to integer overflow.

GO ON TO THE NEXT PAGE.

3. 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

GO ON TO THE NEXT PAGE.

Question 2: APLine

Intent: *Design complete* `APLine` *class including constructor,* `getSlope` *and* `isOnLine` *methods*

+1 Complete, correct header for `APLine` [`class APLine`]
Note: Accept any visibility except private

+1 1/2 State maintenance
- **+1/2** Declares at least one instance variable capable of maintaining numeric value
- **+1/2** Declares at least three instance variables capable of maintaining numeric values
- **+1/2** All state variables have private visibility

Note: Accept any numeric type (primitive or object)
Note: Accept any distinct Java-valid variable names

+1 1/2 `APLine` Constructor
Method header
- **+1/2** Correctly formed header (visibility not private; name `APLine`)
- **+1/2** Specifies exactly three numeric parameters

Method body
- **+1/2** Sets appropriate state variables based on parameters (no shadowing errors)

Note: Interpret instance fields by usage not by name

+2 1/2 `getSlope`
Method header
- **+1/2** Correct method header
(visibility not private; type `double` or `Double`; name `getSlope`; parameterless)

Method body
- **+1/2** Computation uses correct formula for slope
- **+1** Computation uses double precision (no integer division)
- **+1/2** Returns computed value

+2 1/2 `isOnLine`
Method header
- **+1/2** Correct formed header (visibility not private; type `boolean` or `Boolean`, name `isOnLine`)
- **+1/2** Specifies exactly two numeric parameters

Method body
- **+1/2** Computation uses correct formula involving state and parameters (`a*x + b*y + c`)
- **+1/2** Computation uses correct comparison test (equal to zero)
- **+1/2** Returns `true` if is on this `APLine`; `false` otherwise

AP® COMPUTER SCIENCE A
2010 CANONICAL SOLUTIONS

Question 2: APLine

```
public class APLine {
  /** State variables. Any numeric type; object or primitive. */
  private int a, b, c;

  /** Constructor with 3 int parameters. */
  public APLine(int a, int b, int c) {
    this.a = a;
    this.b = b;
    this.c = c;
  }

  /** Determine the slope of this APLine. */
  public double getSlope() {
    return ( - (this.a / (double) this.b));
  }

  /** Determine if coordinates represent a point on this APLine. */
  public boolean isOnLine(int x, int y) {
    return (0 == (this.a * x) + (this.b * y) + this.c);
  }
}


// Alternative solution (state variables of type double):

public class APLine {
  private double a1, b1, c1;

  public APLine(int a, int b, int c) {
    this.a1 = a;
    this.b1 = b;
    this.c1 = c;
  }

  public double getSlope() {
    return -(this.a1 / this.b1);
  }

  public boolean isOnLine(int x, int y) {
    return (0 == (this.a1 * x) + (this.b1 * y) + this.c1);
  }
}
```

These canonical solutions serve an expository role, depicting general approaches to a solution. Each reflects only one instance from the infinite set of valid solutions. The solutions are presented in a coding style chosen to enhance readability and facilitate understanding.