

4. This question involves a path through a two-dimensional (2D) array of integers, where the path is based on the values of elements in the array. When an element of the 2D array is accessed, the first index is used to specify the row and the second index is used to specify the column. The following `Location` class represents a row and column position in the 2D array.

```
public class Location
{
    private int theRow;
    private int theCol;

    public Location(int r, int c)
    {
        theRow = r;
        theCol = c;
    }

    public int getRow()
    { return theRow; }

    public int getCol()
    { return theCol; }
}
```

GO ON TO THE NEXT PAGE.

The following `GridPath` class contains the 2D array and methods to use to determine a path through the array. You will write two methods of the `GridPath` class.

```
public class GridPath
{
    /** Initialized in the constructor with distinct values that never change */
    private int[][] grid;

    /**
     * Returns the Location representing a neighbor of the grid element at row and col,
     * as described in part (a)
     * Preconditions: row is a valid row index and col is a valid column index in grid.
     * row and col do not specify the element in the last row and last column of grid.
     */
    public Location getNextLoc(int row, int col)
    { /* to be implemented in part (a) */ }

    /**
     * Computes and returns the sum of all values on a path through grid, as described in
     * part (b)
     * Preconditions: row is a valid row index and col is a valid column index in grid.
     * row and col do not specify the element in the last row and last column of grid.
     */
    public int sumPath(int row, int col)
    { /* to be implemented in part (b) */ }

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

GO ON TO THE NEXT PAGE.

- (a) Write the `getNextLoc` method, which returns a `Location` object that represents the smaller of two neighbors of the grid element at `row` and `col`, according to the following rules.
- The two neighbors that are considered are the element below the given element and the element to the right of the given element, if they exist.
 - If both neighbors exist, the `Location` of the neighbor with the smaller value is returned. Two neighbors will always have different values.
 - If only one neighbor exists, the `Location` of the existing neighbor is returned.

For example, assume that `grid` contains the following values.

	0	1	2	3	4
0	12	3	4	13	5
1	11	21	2	14	16
2	7	8	9	15	0
3	10	17	20	19	1
4	18	22	30	25	6

The following table shows some sample calls to `getNextLoc`.

Method Call	Explanation
<code>getNextLoc(0, 0)</code>	Returns the neighbor to the right (the <code>Location</code> representing the element at row 0 and column 1), since $3 < 12$
<code>getNextLoc(1, 3)</code>	Returns the neighbor below (the <code>Location</code> representing the element at row 2 and column 3), since $15 < 14$
<code>getNextLoc(2, 4)</code>	Returns the neighbor below (the <code>Location</code> representing the element at row 3 and column 4), since the given element has no neighbor to the right
<code>getNextLoc(4, 3)</code>	Returns the neighbor to the right (the <code>Location</code> representing the element at row 4 and column 4), since the given element has no neighbor below

In the example, the `getNextLoc` method will never be called with row 4 and column 4, as those values would violate the precondition of the method.

GO ON TO THE NEXT PAGE.

Complete the getNextLoc method.

```
/**
 * Returns the Location representing a neighbor of the grid element at row and col,
 * as described in part (a)
 * Preconditions: row is a valid row index and col is a valid column index in grid.
 * row and col do not specify the element in the last row and last column of grid.
 */
public Location getNextLoc(int row, int col)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

Class information for this question

```
public class Location
private int theRow
private int theCol

public Location(int r, int c)
public int getRow()
public int getCol()

public class GridPath
private int[][] grid

public Location getNextLoc(int row, int col)
public int sumPath(int row, int col)
```

GO ON TO THE NEXT PAGE.

- (b) Write the `sumPath` method, which returns the sum of all values on a path in `grid`. The path begins with the element at `row` and `col` and is determined by successive calls to `getNextLoc`. The path ends when the element in the last row and the last column of `grid` is reached.

For example, consider the following contents of `grid`. The shaded elements of `grid` represent the values on the path that results from the method call `sumPath(1, 1)`. The method call returns 19 because $3 + 2 + 9 + 4 + 0 + 1 = 19$.

	0	1	2	3	4
0	12	30	40	25	5
1	11	3	22	15	43
2	7	2	9	4	0
3	8	33	18	6	1

GO ON TO THE NEXT PAGE.

Write the `sumPath` method. Assume `getNextLoc` works as intended, regardless of what you wrote in part (a). You must use `getNextLoc` appropriately in order to receive full credit.

```
/**
 * Computes and returns the sum of all values on a path through grid, as described in
 * part (b)
 * Preconditions: row is a valid row index and col is a valid column index in grid.
 * row and col do not specify the element in the last row and last column of grid.
 */
public int sumPath(int row, int col)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

Class information for this question

```
public class Location
private int theRow
private int theCol

public Location(int r, int c)
public int getRow()
public int getCol()

public class GridPath
private int[][] grid

public Location getNextLoc(int row, int col)
public int sumPath(int row, int col)
```

GO ON TO THE NEXT PAGE.

Question 4: 2D Arrays**9 points****Canonical solution**

- (a) `public Location getNextLoc(int row, int col)` **3 points**
- ```
{
 if (row == grid.length - 1)
 {
 return new Location(row, col + 1);
 }
 else if (col == grid[0].length - 1)
 {
 return new Location(row + 1, col);
 }
 else if (grid[row + 1][col] < grid[row][col + 1])
 {
 return new Location(row + 1, col);
 }
 else
 {
 return new Location(row, col + 1);
 }
}
```
- (b) `public int sumPath(int row, int col)` **6 points**
- ```
{
    int sum = 0;

    while (row < grid.length - 1 || col < grid[0].length - 1)
    {
        sum += grid[row][col];

        Location loc = getNextLoc(row, col);
        row = loc.getRow();
        col = loc.getCol();
    }
    return sum + grid[row][col];
}
```

(a) `getNextLoc`

Scoring Criteria		Decision Rules	
1	Guards against out-of-bounds access of <code>grid</code> elements	<p>Responses can still earn the point even if they</p> <ul style="list-style-type: none"> fail to access any element of <code>grid</code> in this part, as long as the guard prevents the returned <code>Location</code> from being out of bounds <p>Responses will not earn the point if they</p> <ul style="list-style-type: none"> return a <code>Location</code> that would be out of bounds 	1 point
2	Accesses both an element of <code>grid</code> to the right and an element of <code>grid</code> below <code>row</code> and <code>col</code>	<p>Responses can still earn the point even if they</p> <ul style="list-style-type: none"> access elements of <code>grid</code> out of bounds <p>Responses will not earn the point if they</p> <ul style="list-style-type: none"> fail to access elements of <code>grid</code> correctly 	1 point
3	Returns <code>Location</code> of appropriate grid element (<i>algorithm</i>)	<p>Responses can still earn the point even if they</p> <ul style="list-style-type: none"> incorrectly guard against out-of-bounds access of grid elements <p>Responses will not earn the point if they</p> <ul style="list-style-type: none"> call the <code>Location</code> constructor incorrectly fail to consider all four cases 	1 point
Total for part (a)			3 points

(b) `sumPath`

Scoring Criteria		Decision Rules	
4	Initializes and increases variable to store sum of grid values	<p>Responses can still earn the point even if they</p> <ul style="list-style-type: none"> fail to initialize a local variable in a recursive solution, as long as an element of the grid is added to the recursive call <p>Responses will not earn the point if they</p> <ul style="list-style-type: none"> initialize the variable to something other than <code>0</code> or an element of grid increment the sum variable using something other than an element of grid 	1 point
5	Determines the path based on successive calls to <code>getNextLoc</code> while current position is not the bottom-right position of grid (<i>no bounds errors</i>) (<i>algorithm</i>)	<p>Responses can still earn the point even if they</p> <ul style="list-style-type: none"> fail to access an element of <code>grid</code> call <code>getNextLoc</code> incorrectly access row/column of next location incorrectly <p>Responses will not earn the point if they</p> <ul style="list-style-type: none"> fail to call <code>getNextLoc</code> fail to use row/column derived from <code>getNextLoc</code> return value in subsequent calls stop loop early (omit required path locations) or late (violate <code>getNextLoc</code> precondition) due to incorrect boundary condition 	1 point
6	Calls <code>getNextLoc</code> (<i>in the context of a loop</i>)	<p>Responses can still earn the point even if they</p> <ul style="list-style-type: none"> call <code>getNextLoc</code> within an incorrect loop <p>Responses will not earn the point if they</p> <ul style="list-style-type: none"> call <code>getNextLoc</code> on the class or on an object other than <code>this</code> (use of <code>this</code> is optional) fail to call <code>getNextLoc</code> with two <code>int</code> arguments 	1 point
7	Calls <code>getRow</code> and <code>getCol</code> on a <code>Location</code> object	<p>Responses will not earn the point if they</p> <ul style="list-style-type: none"> call either method incorrectly 	1 point

8	Accesses a <code>grid</code> element at positions derived from the call to the next location method	Responses can still earn the point even if they <ul style="list-style-type: none">access an incorrect <code>grid</code> elementonly access the grid at <code>row</code> and <code>col</code>, if the solution is recursive and the parameters of the recursive call are derived from a call to the next location method	1 point
9	Computes sum of values along path (<i>algorithm</i>)	Responses can still earn the point even if they <ul style="list-style-type: none">stop loop early or late due to incorrect boundary conditionfail to return the computed sum (<i>return not assessed in this part</i>) Responses will not earn the point if they <ul style="list-style-type: none">fail to include the first or last visited location in the sum	1 point
Total for part (b)			6 points
Total for question 4			9 points