AP® COMPUTER SCIENCE A
2013 SCORING GUIDELINES

Question 4: SkyView

Part (a)	`SkyView` constructor	**5 points**

Intent: *Construct* `SkyView` *object from 1D array of scan data*

+1 Constructs correctly-sized 2D array of doubles and assigns to instance variable `view`

+1 Initializes at least one element of `view` with value from element of `scanned` (*must be in context of loop*)

+1 Places consecutive values from `scanned` into at least one row of `view` in original order

+1 Places consecutive values from `scanned` into at least one row of `view` in reverse order

+1 On exit: all elements of `view` have correct values (*no bounds errors on* `view` *or* `scanned`)

Part (b)	`getAverage`	**4 points**

Intent: *Compute and return average of rectangular section of* `view`*, specified by parameters*

+1 Declares and initializes a `double` accumulator

+1 Adds all and only necessary values from `view` to accumulator (*no bounds errors*)

+1 Computes average of specified rectangular section

+1 Returns the computed average (*computation must involve* `view`)

Question-Specific Penalties

-2 (v) Consistently uses incorrect array name instead of `view/scanned`

4A a

Complete the `SkyView` constructor below.

```
/** Constructs a SkyView object from a 1-dimensional array of scan data.
 *  @param numRows the number of rows represented in the view
 *         Precondition: numRows > 0
 *  @param numCols the number of columns represented in the view
 *         Precondition: numCols > 0
 *  @param scanned the scan data received from the telescope, stored in telescope order
 *         Precondition: scanned.length == numRows * numCols
 *  Postcondition: view has been created as a rectangular 2-dimensional array
 *                 with numRows rows and numCols columns and the values in
 *                 scanned have been copied to view and are ordered as
 *                 in the original rectangular area of sky.
 */
public SkyView(int numRows, int numCols, double[] scanned) {
    view = new double[numRows][numCols];
    int k = 0; // scanned index

    for (int r = 0; r < numRows; r++) {
        if (r % 2 == 0) {
            for (int c = 0; c < numCols; c++) {
                view[r][c] = scanned[k];
                k++;
            }
        }
        else {
            for (int c = numCols - 1; c >= 0; c--) {
                view[r][c] = scanned[k];
                k++;
            }
        }
    }
}
```

Part (b) begins on page 24.

4Ab

Complete method `getAverage` below.

```
/** Returns the average of the values in a rectangular section of view.
 *  @param startRow the first row index of the section
 *  @param endRow the last row index of the section
 *  @param startCol the first column index of the section
 *  @param endCol the last column index of the section
 *  Precondition: 0 <= startRow <= endRow < view.length
 *  Precondition: 0 <= startCol <= endCol < view[0].length
 *  @return the average of the values in the specified section of view
 */
public double getAverage(int startRow, int endRow,
                         int startCol, int endCol) {
    int width = endCol - startCol + 1;
    int height = endRow - startRow + 1;
    int numValues = width * height;
    int sum = 0;

    for (int r = startRow; r <= endRow; r++) {
        for (int c = startCol; c <= endCol; c++) {
            sum += view[r][c];
        }
    }

    return sum / numValues;
}
```


GO ON TO THE NEXT PAGE.

4Ba

Complete the `SkyView` constructor below.

```
/** Constructs a SkyView object from a 1-dimensional array of scan data.
 *  @param numRows  the number of rows represented in the view
 *          Precondition: numRows > 0
 *  @param numCols  the number of columns represented in the view
 *          Precondition: numCols > 0
 *  @param scanned  the scan data received from the telescope, stored in telescope order
 *          Precondition: scanned.length == numRows * numCols
 *  Postcondition: view has been created as a rectangular 2-dimensional array
 *                 with numRows rows and numCols columns and the values in
 *                 scanned have been copied to view and are ordered as
 *                 in the original rectangular area of sky.
 */
public SkyView(int numRows, int numCols, double[] scanned)
{
  double[][] temp = new double[numRows][numCols];
    int count=0;
  for (int x=0; x<numRows; x++)
  {
    for (int y=0; y<numCols; y++)
    { if (x==0||x%2==0)
      {
        temp[x][y] = scanned[count];
      }
      else
      {
        temp[x][y+numCols-y-1]=scanned[count];
      }
      count++;
    }
  }
  view = temp;
}
```

Part (b) begins on page 24.

4Bb

Complete method `getAverage` below.

```
/** Returns the average of the values in a rectangular section of view.
 *  @param startRow the first row index of the section
 *  @param endRow the last row index of the section
 *  @param startCol the first column index of the section
 *  @param endCol the last column index of the section
 *  Precondition: 0 <= startRow <= endRow < view.length
 *  Precondition: 0 <= startCol <= endCol < view[0].length
 *  @return the average of the values in the specified section of view
 */
public double getAverage(int startRow, int endRow,
                         int startCol, int endCol)
{ int count = 0;
  int sum = 0;
  for (int x = startRow; x <= endRow; x++)
  {
    for (int y = startCol; y <= endCol; y++)
    {
      sum = sum + view[x,y];
      count++;
    }
  }
  int average = sum / count;
  return average
}
```


GO ON TO THE NEXT PAGE.

4 Ca

Complete the `SkyView` constructor below.

```
/** Constructs a SkyView object from a 1-dimensional array of scan data.
 *  @param numRows  the number of rows represented in the view
 *                  Precondition: numRows > 0
 *  @param numCols  the number of columns represented in the view
 *                  Precondition: numCols > 0
 *  @param scanned  the scan data received from the telescope, stored in telescope order
 *                  Precondition: scanned.length == numRows * numCols
 *  Postcondition: view has been created as a rectangular 2-dimensional array
 *                 with numRows rows and numCols columns and the values in
 *                 scanned have been copied to view and are ordered as
 *                 in the original rectangular area of sky.
 */
public SkyView(int numRows, int numCols, double[] scanned)
```

```
{
  for (int r=0; r<numRow; r++)
  { if (r%2==0)
    for (int c=0; c<numCol; r++)
    {
      view[r][c] = scanned[t]
    }
  }
  else
  {
    for (int c2=numCol; c2>0; c2++)
    {
      view[r][c2] = scanned[t];
    }
  }
}
}
```

Part (b) begins on page 24.

GO ON TO THE NEXT PAGE.

4 a b

Complete method `getAverage` below.

```
/** Returns the average of the values in a rectangular section of view.
 *  @param startRow the first row index of the section
 *  @param endRow the last row index of the section
 *  @param startCol the first column index of the section
 *  @param endCol the last column index of the section
 *  Precondition: 0 <= startRow <= endRow < view.length
 *  Precondition: 0 <= startCol <= endCol < view[0].length
 *  @return the average of the values in the specified section of view
 */
public double getAverage(int startRow, int endRow,
                         int startCol, int endCol)
{
  double store=0.0;
  for (int r=startRow; r<endRow; r++)
  {
    for (int c=startCol; c<endCol; c++)
    {
      store+= view[r][c];
    }
  }

  int RC= endRow-startRow;
  int CC= endCol-startCol;
  double avg= store/(RC+CC);

  return avg;
}
```


GO ON TO THE NEXT PAGE.

AP® COMPUTER SCIENCE A
2013 SCORING COMMENTARY

Question 4

Overview

This question addressed the construction, initialization and use of a rectangular two-dimensional array of primitive values, accessing array elements, managing an accumulator, and returning a value. Students were asked to implement a constructor and method of the `SkyView` class. In part (a) students were required to implement a constructor, which created a rectangular array (dimensions determined by parameters `numRows` and `numCols`) and initialized the instance array with values from a 1D parameter array. The mapping of the 1D parameter array into the 2D array was defined as alternating the direction of the fill of each row so that the first row must fill from left to right and the second from left to right and so on. In part (b) students were required to implement the method `computeAverage`, which used parameters `startRow`, `endRow`, `startCol`, `endCol` to define a rectangular region over which to compute the average. The method must create a `double` accumulator initialized to 0.0 and then traverse the relevant section of the 2D array, using an outer loop starting at `startRow`, and ending at `endRow`, and an inner loop starting at `startCol` and ending at `endCol`. Within the loops, each element is added to the sum. After the loops complete, the sum is divided by the number of elements to obtain the average. The average is returned by the method.

Sample: 4A
Score: 8

In part (a), a correctly-sized two-dimensional array of `double` values is created and assigned to the instance variable `view`. Nested loops are used to assign elements of `scanned` to the appropriate elements of `view`. An outer loop handles each row, and separate inner loops are used, based on whether a row is even or odd, to place consecutive values from `scanned` into rows of `view` in the original order and in reverse order, respectively. Upon exit, all elements of `view` have correct values. Part (a) earned 5 points.

In part (b), the accumulator is declared as an `int` instead of as a `double`. The nested loops ensure that all of the necessary values from `view`, and only those values, are added to the accumulator. The size of the specified rectangular section is correctly computed, resulting in a correct computation of the average, which is returned. Part (b) earned 3 points.

Sample: 4B
Score: 6

In part (a), a correctly-sized two-dimensional array of `double` values is constructed and assigned to the instance variable `view`. At least one element of `view` is initialized with a value from an element of `scanned`, and consecutive values from `scanned` are placed into at least one row of `view` in original order. The extraneous "`y +`" causes all values to be assigned to the same element of `view` in odd-numbered rows, however, so not all elements of `view` contain correct values. Part (a) earned 3 points.

In part (b), all variables are declared as `int` instead of `double`. The nested loops ensure that all of the necessary values from `view`, and only those values, are added to `sum`. A counter is used to keep track of the size of the specified rectangular section. The average is correctly computed and returned. Part (b) earned 3 points.

Question 4 (continued)

Sample: 4C
Score: 2

In part (a), there is no construction of a two-dimensional array, and no element of `scanned` is assigned to `view`. Part (a) earned 0 points.

In part (b), a `double` accumulator is declared and initialized. Only one element from the specified rectangular section of `view` is accumulated because the loop variables, `r` and `c`, are never incremented. The divisor is calculated incorrectly, but the computed average is returned. Part (b) earned 2 points.