› Answers and Explanations

1. The answer is C.

- This is not a nested `for` loop, which means each element of the 2D array will not be accessed.
- The first element that is accessed would be the last row, 1st column, table [length - 0 - 1] [0].
- The second element that is accessed would be the next to last row and the 2nd column. Table[length - 1 - 1] [1].
- The loop will continue moving upwards along the minor diagonal until all elements are accessed and added to sum.

2. The answer is B

- This uses a nested `for` loop with both loop control variables starting at 0 and ending at the length of the row or column, which means each element of the 2D array is potentially accessed.
- It tests each element against 0. If that element is negative (< 0), it will be changed to the opposite, which is positive.
- After the loops are complete, each element in the table will be nonnegative (positive or 0).

3. The answer is D.

- Only one row needs to be summed, so a single `for` loop is appropriate.
- Since row n is the row to be summed, this value needs to remain unchanged throughout the loop.
- Choice D correctly accesses each element in the correct row in `mat`.

4. The answer is D.

- Choice A would be the result if both loops ended at the length of the table instead of length-1.
- Choice B would be the result if both loops ended at the length of the table instead of length-1 and the operation was addition (+) and not multiplication (*).
- Choice C would be the result if the operation was addition (+) and not multiplication (*).
- Choice E would be the result if the loop started at the last row.

5. We can find the even rows by using the condition (row % 2 == 0). In other words, if I can divide the row number by 2 and not have a remainder, it's an even row. Even rows just get copied as is. Odd rows get copied from the row above [row − 1].

Here is the completed method.

```
public int[][] modify(int[][] arr) {

    int[][] newArr = new int[arr.length][arr[0].length];

    for (int row = 0; row < arr.length; row++)
    {
        for (int col = 0; col < arr[row].length; col++)
        {
            if (row % 2 == 0)
            {
                newArr[row][col]  = arr[row][col];           // even row
            }
            else
            {
                newArr[row][col]  = arr[row - 1][col];       // odd row
            }
        }
    }
    return newArr;
}
```

6. There are many ways to solve this problem. This implementation uses the fact that grid[row][col] is true when row + col is even.

```
public static boolean[][] makeGrid(int rows, int cols)
{
    boolean[][] grid;
    grid = new boolean[rows][cols];        //initializes all elements to false
    for (int r = 0; r < rows; r++)
    {
        for (int c = 0; c < cols; c++)
        {
            if ((r + c) % 2 == 0)          // r + c is even
            {
                grid[r][c] = true;
            }
        }
    }
    return grid;
}
```

Here is a tester class so you can see if the way you wrote the method was also correct. Copy in your code. Try it with all shapes and sizes of grid to test it thoroughly.

```
public class GridBuilder
{
    public static void main(String[] args)
    {
        boolean[][] grid = makeGrid(3, 4); //change (3,4) to test
        for (int r = 0; r < grid.length; r++)
        {
            for (int c = 0; c < grid[0].length; c++)
            {
                System.out.print(grid[r][c] + "\t");
            }
            System.out.println();
        }
    }

    public static boolean[][] makeGrid(int rows, int cols)
    {
        boolean[][] grid;
        //put your code in here
    }
}
```

7. Find all the songs that contain the word "Love" in the title.

Algorithm:
Step 1: Initialize a count variable to zero
Step 2: Look at each of the scores in the 2D array (using Row-Major order)
Step 3: If the song title contains the target String, then increment the count
Step 4: Continue until you reach the end of the list
Step 5: Return count

Pseudocode:

```
for (iterate through all the rows in the 2D array)
{
    for (iterate through all the columns in the 2D array)
    {
        if (list[row][column] contains target String)
        {
            count = count + 1
        }
    }
}
return count
```

Java code:

```
public static int findCount(String[][] arr, String target)
{
   int count = 0;
   for (int r = 0; r < arr.length; r++)
   {
      for (int c = 0; c < arr[r].length; c++)
      {
         if (arr[r][c].indexOf(target) != -1)      // target is in title
         {
            count++;
         }
      }
   }
   return count;
}
```