HW – 2D Array

1. StarTable.java: Ask users for the dimension a table. Create a 2D array of characters with the specified dimensions, and initialize each element to be '*'. Then print out the content of the array as a table.

2. TimesTable.java: Write a program that generates a 12 by 12 multiplication table. It stores the table in a 2 dimensional array of integers and finally output the contents of the array in the form of a formatted table (with the header row and column) to the screen, as shown. Please note the dimension of the 2D array is 12 by 12 (the header row and column are not part of the array) **Make sure the table is formatted as a multiplication table (each row of numbers is right under the preceding row); `printf` may be helpful with the formatting.**

   Sample output:

   ```
        1    2    3    4    5    6    7    8    9    10   11   12
   1    1    2    3    4    5    6    7    8    9    10   11   12
   2    2    4    6    8    10   12   14   16   18   20   22   24
   3    3    6    9    12   15   18   21   24   27   30   33   36
   4    4    8    12   16   20   24   28   32   36   40   44   48
   5    5    10   15   20   25   30   35   40   45   50   55   60
   6    6    12   18   24   30   36   42   48   54   60   66   72
   7    7    14   21   28   35   42   49   56   63   70   77   84
   8    8    16   24   32   40   48   56   64   72   80   88   96
   9    9    18   27   36   45   54   63   72   81   90   99   108
   10   10   20   30   40   50   60   70   80   90   100  110  120
   11   11   22   33   44   55   66   77   88   99   110  121  132
   12   12   24   36   48   60   72   84   96   108  120  132  144
   ```

   MEMORY TRACES ON NEXT PAGE

3. Perform the following memory trace:

```
final int ROWS = 2;
final int COLS = 3;
int count = 0;
int[][] myArray = new int[ROWS][COLS]

for (int row = 0; row < ROWS; row++)
{
     for (int col = 0; col < COLS; col++)
     {
          myArray[row][col] = count;
          count++;
     }
     count += 10;
}
```

ROWS (final int):	2

COLS (final int):	3

count (int):	~~0 1 2 12 13~~ 14

myArray (int[ ][ ]):	0	1	2
	12	13	14

---

row (int):	~~0 1~~ 2

col (int):	~~0 1 2 3~~ ~~0 1 2~~ 3

4. Perform the following memory trace. Note that the loops have switched places:

```
final int ROWS = 2;
final int COLS = 3;
int count = 0;
int[][] myArray = new int[ROWS][COLS]

for (int col = 0; col < COLS; col++)
{
     for (int row = 0; row < ROWS; row++)
     {
          myArray[row][col] = count;
          count++;
     }
     count += 10;
}
```

ROWS (final int):	2

COLS (final int):	3

count (int):	~~0 1~~ ~~11 12~~ ~~22~~ 23

myArray (int[ ][ ]):	0	11	22
	1	12	23

count (int):	

row (int):	~~0 1 2~~ ~~0 1 2~~ ~~0 1~~ 2

col (int):	~~0 1 2~~ 3