1. **Create** and **initialise** a two-dimensional **int** array with 10 rows and 15 columns. Each element in the array is to be initialised to a random value in the range 1 – 100. However, you must ensure the following.

    - All elements in row 2 must be multiples of 2.
    - All elements in row 3 must be multiples of 3.
    - All elements in row 4 must be multiples of 4 etc..

    There is no restriction on the values that can be stored in row 0 or row 1. You can just initialise the elements in these rows to values in the range 1 – 100.

    You must create a method to achieve this task. The signature of the method is to be as follows:

    ```
    private static int[][] initArray(int row, int col) { … }
    ```

    Where **row** is the number of rows in the array and **col** is the number of columns in the array. Once the method has completed, it is to **return** the newly created and initialised array.

    (30 Marks)

---

2. **Print** the array neatly to the screen (in rows of 15 columns). You must create a method to achieve this task. The signature of the method is to be as follows:

    ```
    public static void print(int[][] array) { … }
    ```

    Where **array** is the array to be printed.

    (5 Marks)

---

3. Write a method with the following signature:

```
private static void printAdjacentSum(int x, int[][] array) { …}
```

This method must print the indexes of any adjacent numbers in a given row of the array that sum to x. If no adjacent numbers in the array sum to x, then print nothing.

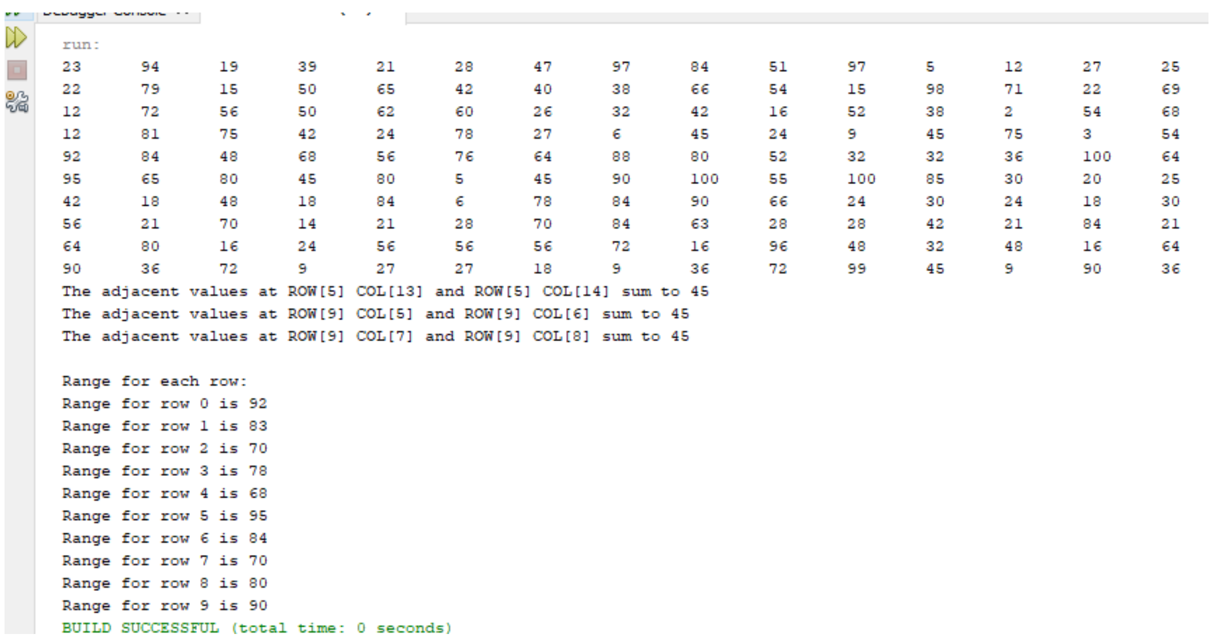
(35 Marks)

---

4. For every row in the array you must calculate and display the range. The range is a simple statistical measure which calculates the difference between the lowest and highest values (in a list of numbers). For example if the largest value in a list is 21 and the smallest is 13 the range is 8 (21 – 13 = 8). You must create a method to achieve this task, and it must accept at least one argument, a **row** of the 2D array (you can pass in more arguments if you wish). When displaying the range, you must also display the row number (see the output from my solution below).

(30 Marks)

Your output should look something like the following:



```
run:
23      94      19      39      21      28      47      97      84      51      97      5       12      27      25
22      79      15      50      65      42      40      38      66      54      15      98      71      22      69
12      72      56      50      62      60      26      32      42      16      52      38      2       54      68
12      81      75      42      24      78      27      6       45      24      9       45      75      3       54
92      84      48      68      56      76      64      88      80      52      32      32      36      100     64
95      65      80      45      80      5       45      90      100     55      100     85      30      20      25
42      18      48      18      84      6       78      84      90      66      24      30      24      18      30
56      21      70      14      21      28      70      84      63      28      28      42      21      84      21
64      80      16      24      56      56      56      72      16      96      48      32      48      16      64
90      36      72      9       27      27      18      9       36      72      99      45      9       90      36
The adjacent values at ROW[5] COL[13] and ROW[5] COL[14] sum to 45
The adjacent values at ROW[9] COL[5] and ROW[9] COL[6] sum to 45
The adjacent values at ROW[9] COL[7] and ROW[9] COL[8] sum to 45

Range for each row:
Range for row 0 is 92
Range for row 1 is 83
Range for row 2 is 70
Range for row 3 is 78
Range for row 4 is 68
Range for row 5 is 95
Range for row 6 is 84
Range for row 7 is 70
Range for row 8 is 80
Range for row 9 is 90
BUILD SUCCESSFUL (total time: 0 seconds)
```

NOTE.

In the screen grab above, I was testing that adjacent values in the array were equal to 45.

All of the methods you create must be called from **main.**

Including superfluous lines of code in your solution will see your mark reduced. For example, including 100 lines of code where 5 would suffice.

Any projects that contain syntax errors will receive a mark of 0.

Make sure you upload a zipped Netbeans project to Moodle.