CS 200 Homework Assignment 1

Given: 12 September 2020

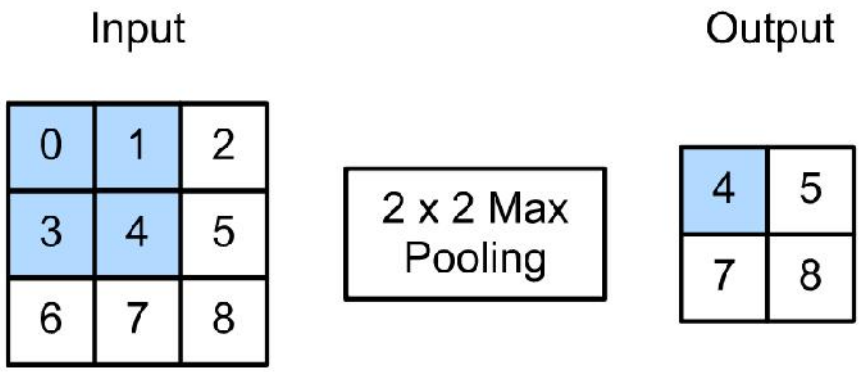
Due: 26 September 2020 @ 11:59 PM

100 points (+ 20 points of extra credit) available

This assignment involves the following Java topics:

1. User-defined methods
2. Loops
3. Arrays
4. Input-output

In this assignment, you will create a two-dimensional array and perform some calculations inside some sliding windows based on this array. These types of functions occur regularly in convolutional neural networks (CNNs) as part of deep learning/machine learning. See below for an example for one such function called a max pooling layer.



In reality, it is actually more complex than this, but for this assignment, we will start with the basics.

A. Creating the initial array

You will prompt the user for an integer value representing the **dimension size** of the array (an integer value from 2 to 8, inclusive). The array you create will always be a square (i.e. the same number of rows and columns). For example, if the user provides a 5, a 5x5 array will be created. You need to check to see if the number is between 2 and 8, inclusive and return an error message if the number supplied is not within this range. You can assume an integer value will be provided and no need to check for this.

Next, you will prompt the user for a **seed** for the random number generator. You can assume an integer value and no error checking is needed for the seed value.

The random number generator will use the seed to populate the array with random integers between 0 and 15, inclusive.

You will create a method and pass these two integers (the seed and the array size) to a method you create. Once the array is populated with random numbers in your method, you will print out the array. The array

must be created within a separate method (not in main()) and must be passed back to the main method after it is created.

B. Printing the initial array

Print out the resulting array with a label (e.g., Main array"). You can either print out the array in two-digit format in the method used to create the method or print it in your main() method. For an array of size 6 using the seed value of 7, the 6x6 array will look like the following:

```
Main array:
11 10 11 00 05 07
14 11 11 09 05 15
01 04 13 11 01 10
14 01 10 01 12 11
09 11 07 09 04 09
06 09 12 09 03 11
```

Note the two-digit format for the numbers with the leading "0" for numbers less than 10 and the space between elements. This keeps things aligned correctly. Hint: It is helpful to use printf() to perform this formatting, along with println().

C. A sliding window to calculate sums.

Once your initial array has been created, printed, and returned as a parameter to main(), a separate method will be called. This method will use a sliding window to calculate the sums within the sliding window. You can receive full credit by limiting your method to using a 2x2 sliding window, but an extra credit opportunity exists to prompt the user for an integer to create various n x n window sizes as supplied by the user through an input prompt.

The window will move over by one each time (called a *stride* of 1). Once it hits the last window that fits in the first row, it will go down a row and move across the array.

For example, if our main array looks like the following:

```
Main array:
11 10 11 00 05 07
14 11 11 09 05 15
01 04 13 11 01 10
14 01 10 01 12 11
09 11 07 09 04 09
06 09 12 09 03 11
```

It will begin by taking the values in the 2x2 window in the upper left corner and summing the 4 values in the window. For example, it will first evaluate the 2x2 window of:

```
11 10
14 11
```

which represents

```
mainArray[0][0],
mainArray[0][1],
mainArray[1][0], and
mainArray[1][1]
```

which sums to 46. Next it will slide over 1 value and calculate the sum of the next 4 values (for a 2x2 window, two of these values will also appear in the previous calculation

```
10 11
11 11
```

which represents

```
mainArray[0][1],
mainArray[0][2],
mainArray[1][1], and
mainArray[1][2].
```

Do you see a pattern here?

Your method will store the sum of these 4 numbers in a separate two-dimensional array and print it out. You can either print it out in the method you create, or return it and print it from main(). Print out the resulting array with a label (e.g., Sum array:") The results from the previous 6x6 array would be the following:

```
Sum array:
46 43 31 19 32
30 39 44 26 31
20 28 35 25 34
35 29 27 26 36
35 39 37 25 27
```

Note that it is a 5x5 array. Recall that each dimension of the array will be n-k+1 where n is the dimension of the array (in this case, 6) and k is the window size (in this case, 2).

D. A sliding window to calculate averages.

Next, create a separate method that accepts the array as a parameter and calculates the average of each window using the same approach used for calculating sums. If you can get the sums to work, calculating the averages requires only a small modification. Print out the resulting array with a label (e.g., Avg array:")

For example, if our main array looks like the following:

```
Main array:
11 10 11 00 05 07
14 11 11 09 05 15
01 04 13 11 01 10
14 01 10 01 12 11
09 11 07 09 04 09
06 09 12 09 03 11
```

The averages array will look like the following:

```
Avg array:
11 10 07 04 08
07 09 11 06 07
05 07 08 06 08
08 07 06 06 09
08 09 09 06 06
```

Notice that it rounds down to the nearest whole integer.

E. A sliding window to calculate minimum value.

Next, create a separate method that accepts the main array as a parameter and outputs the single minimum value in each sliding window. You may use the same general nesting loop approach used for calculating sums and averages. There is a min() method in the Java Math library that will be useful here but it only compares two values at a time and the 2x2 sliding window will have four. Print out the resulting array with a label (e.g., Min array:") Using the same example main array, our output would be:

```
Min array:
10 10 00 00 05
01 04 09 01 01
01 01 01 01 01
01 01 01 01 04
06 07 07 03 03
```

F. A sliding window to calculate maximum value.

Similar to the minimum value method, this uses a separate method to output the maximum value in our sliding window. There is a max() method in the Java Math library. This will need to be output in the following format (use a label to indicate what the output represents such as "Max array")

```
Max array:
14 11 11 09 15
14 13 13 11 15
14 13 13 12 12
14 11 10 12 12
11 12 12 09 11
```

Grading:

No points will be taken off if you do not document your code for this assignment, but 5 points will be deducted if you don't have a brief comment section at the top of your code with your name, the date, and the assignment name ("CS 200 programming assignment 1").

Up to 5 points will be deducted for using cryptic variable names, so make sure your variable names represent the variable's true purpose. For counter variables, using letters like i and j are okay.

Late penalties apply to this homework assignment. Remember the plagiarism policy for the course and for the university.

You are <u>not</u> required to make a narrated video demonstrating your code for this assignment. It will be helpful to get familiar with software like screencastomatic before the next homework assignment as it will be required for the next assignment.

Grading will be done using the following rubric:

A. Creating the initial array (40 points)
    1. Including, prompting the user for the right values
    2. Error checking the input for these values within the correct range
    3. Use of a method to create an array and return the array back to main
B. Printing the initial array (10 points)
    1. Array has the correct format
    2. Array has a label ("Main array") or ("Initial array")
C. A sliding window to calculate sums. (20 points)
    1. Array has the correct format
    2. Array has a label ("Sum array")
    3. Values are correct
D. A sliding window to calculate averages. (5 points)
    1. Array has the correct format
    2. Array has a label ("Avg array")
    3. Values are correct
E. A sliding window to calculate minimum. (20 points)
    1. Array has the correct format
    2. Array has a label ("Min array")
    3. Values are correct
F. A sliding window to calculate maximum. (5 points)
    1. Array has the correct format
    2. Array has a label ("Max array")

3. Values are correct

G. Extra credit (20 points)
   a. Use of various sliding window sizes, up to nxn (prompted to the user)
   b. Error checking on the user inputs
   c. Sum, avg, min, and max calculated correctly

What to turn in:

- Your code, with a commented header containing your name, the assignment name, and the date) contained within a single class, that can be cut and pasted into BlueJ and run after compilation.
- Optional: a narrated video of your code running

Example outputs (you can use it to check your results):

```
Enter the array size (an integer between 2 and 8): 9
That number is outside the range of possible values. Try again.
Enter the array size (an integer between 2 and 8): 5
Enter an integer seed: 373

Main array:
11 13 03 09 01
15 12 07 04 14
04 05 07 02 10
04 03 03 10 15
06 08 11 00 14

Sum array:
51 35 23 28
36 31 20 30
16 18 22 37
21 25 24 39

Avg array:
12 08 05 07
09 07 05 07
04 04 05 09
05 06 06 09

Min array:
11 03 03 01
04 05 02 02
03 03 02 02
03 03 00 00

Max array:
15 13 09 14
15 12 07 14
05 07 10 15
08 11 11 15
```