CSE 11: Accelerated Introduction to Programming

Assignment 4

Loops, Recursion, and Arrays

Due Date: Wednesday, May 1, 11:59 PM

Learning goals:

- Getting practice with loops
- Getting practice with 1D arrays and 2D arrays
- Getting practice with recursion

NOTE: This programming assignment must be done individually.

Your grade will be determined by your most recent submission. If you submit to Gradescope after the deadline, it will be marked late and the late penalty will apply regardless of whether you had past submissions before the deadline.

If your code does not compile on Gradescope and the Autograder fails to execute properly, you will receive an automatic zero on the assignment. Make sure that you do not change the function signatures or import other packages unless otherwise specified.

Coding Style (10 points)

For this programming assignment, we will be enforcing the CSE 11 Coding Style Guidelines. These guidelines can also be found on Canvas. Please ensure to have COMPLETE file headers, class headers, and method headers, to use descriptive variable names and proper indentation, and to avoid using magic numbers.

Part 0: Getting started with the starter code (0 points)

1. If using a personal computer, then ensure your Java software development environment does not have any issues. If there are any issues, then review Assignment 1, or come to the office/lab hours before you start Assignment 4.
2. First, navigate to the `cse11` folder that you have created in Assignment 1 and create a new folder titled `assignment4`

3. Download the starter code. You can download the starter code from Piazza → Resources → Homework → `assignment4.zip`. The starter code should contain two files: `FunWithArrays.java`, and `RecursiveHourglass.java`. Place the starter code within the `assignment4` folder that you have just created

---

Part 1: Implement Methods in `FunWithArrays` (70 points)

In class `FunWithArrays` of the starter code, 9 methods are already declared for you: `findPosition`,`findAvg`, `arrayCopy`, `containsIntegerSequence`, `reverseArray`, `findRange`, `arrayAverage`, `unitTests`, and `main`. **For this part of the assignment, your task is to implement the first seven methods (**`findPosition`,`findAvg`, `arrayCopy`, `containsIntegerSequence`, `reverseArray`, `findRange`, and `arrayAverage`**)**.

`findPosition` (10 points):

This method takes a string array variable, `array`, as the input. You must find the position of the longest string in the array. If there are multiple strings with the maximum number of characters, return the position of the last string in the array with that length. If `array` is `null` or if the length is `0`, then you must return `0`.

Sample:

`findPosition(new String[]{"cat", "mouse", "bear"})` → `1`

`findPosition(new String[]{"cat", "mouse", "bear", "horse", "dog"})` → `3`

Explanation: The maximum length of a string in the array is 5 in both examples. In the first example there is only one string with the length 5, but in the second example there are two strings with length of 5. In the first example we return the position of the only string of length 5. In the second example we return the position of the last string of length 5.

`findAvg` (10 points):

This method takes an integer array variable, `array`, as the input. You must find the average of the integers within this array and return the result as a `double`. Average is defined as the sum of all integers divided by the number of integers. If `array` is `null` or if the length is `0`, then you must return `0.0`.

Sample:

`findAvg(new int[]{1, 2, 3, 4, 5})` → `3.0`

Explanation: The sum of all integers is 15. The number of integers in the array is 5. 15/5 = 3.0.

`arrayCopy` (10 points):

This method takes an integer array variable, `array`, as the input. You must return a **deep copy** of the array with the same contents. A deep copy of the array is defined as an array that has the same contents but is not the same array that is stored in memory. So `arr == arrCopy` must be `false`. If you just returned the same reference that was passed in as an argument, then you will fail the autograder test cases. If `array` is `null`, then you must return `null`. You are **not** allowed to use `System.arraycopy()`.

Sample:

`arrayCopy(new int[]{1, 2, 3, 4, 5})` → `{1, 2, 3, 4, 5}`

`containsIntegerSequence` (10 points):

This method takes an integer array variable, `array`, as the input and returns a `boolean`. It must return `true` if there is a sub array with integers in sequential ascending order; it returns `false` if none of the integers are in sequential ascending order. If `array` is `null`, then you must return `false`. If the length is `0`, you must return `false`.

Sample:

`containsIntegerSequence(new int[]{1, 0, 2})` → `false`

`containsIntegerSequence(new int[]{3, 4, 1, 0, 2})` → `true`

Explanation: The second array is true because it includes the sub array {3, 4}, which is in sequential ascending order. The first array is false because none of the sub arrays: {1,0}, {0,2}, or {1,0,2} are in sequential ascending order.

`reverseArray` (10 points):

This method takes a string array variable, `array`, as the input and does **not return** anything. Instead, it must reverse the array that was passed in as an argument in place. If `array` is `null`, then you do not have to do anything and must simply return from the method.

Sample:
`String[] colors = new String[]{"red", "green", "blue","purple"};`
`reverseArray(colors)` → the same array `colors` that was passed in as an argument will now be: `{"purple", "blue", "green", "red"}`

Be sure that your method modifies the array that was passed in. Making a copy of the array, reversing the copy, then assigning the variable that was passed in to the reference of the copy would fail the autograder.

`findRange` (10 points):

This method takes in one integer 2D array, `array`, as the input and returns an `int` with the range of the array. Range = max_val - min_val. If `array` is null, you must return 0.
Sample:
`findRange(new int[][]{{1, 0, 1}, {2, 4, 1}, {3, 2, 1}})` → `4`

1	0	1
2	4	1
3	2	1

Explanation: Range is the maximum value minus the minimum value. The maximum value in the array is 4, and the minimum value in the array is 0, making the range 4.

`arrayAverage` (10 points)

This method takes in two integer 2D arrays of the same size, `arr1` and `arr2`, as the input and returns a `float` 2D array of the same dimensions, `avg`, where each element in `avg` is the average of the elements in `arr1` and `arr2` at the same location. If the arrays are not the same size or are `null`, then you must return `null`.

Sample:

`findAverage(new int[][]{{1, 0, 1}, {2, 4, 1}, {3, 2, 1}}, {{1, 2, 5}, {6, 3, 1}, {1, 1, 8}})` → `{{1.0, 1.0, 3.0},{4.0, 3.5, 1.0},{2.0, 1.5, 4.5}}`

Part 2: Implement RecursiveHourglass (10 points)

You will be generating a shape, similarly to assignment 2. This time, you will be creating an hourglass, and you will be generating it recursively.

Please take a look at the helper method provided called getLine that you **must** use in your implementation below. This method takes in three parameters, a character `c`, an integer `n`, and another integer `numSpaces`. `getLine` returns a `String` of `n` characters (`c`) with `numSpaces` spaces in the front and back of the line, along with a newline character at the end. Examples will make this clear:

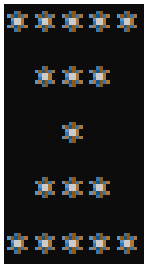
`getLine('@', 3, 0)` → `"@@@\n"`
`getLine('*', 5, 1)` → `" ***** \n"`
`getLine('#', 1, 2)` → `"  #  \n"`

You must NOT change this method and you MUST use it in your implementation below. How you use it is up to you, though.

`recursiveHourglass` (10 points):

This method takes in the three variables as parameters `c`, `height`, and `numSpaces`. This method must generate the exact same structure of the hourglass in Assignment 2 based on the parameters that are passed in. More concretely this method must generate an hourglass that is made up of the char `c` with `height` rows. The first row starts with `n` `c`s and decreases by two each row (one at the front and one at the end). However, the `c`s that were deleted must now be replaced by space characters. When you reach the middle of a single `c`, the hourglass starts to expand again with additional `c`s placed at the beginning and back (with the corresponding spaces deleted). If `height` is less than or equal to zero or if `height` is not an odd number, you must return the empty string (`""`).

For example, if we create an hourglass of height **five** with `c` as `'*'`, then the result string should be: `"*****\n *** \n  *  \n *** \n*****\n"` (where `\n` is the new line character). When this string is printed out, it should look like this:



NOTE: Notice how each row has one extra space in the **beginning and end** until you reach the middle of the hourglass which is just a single `'*'` which then starts reversing. In this example, there are 5 rows.

IMPORTANT (Will lose points if not followed):

1. There must be the right amount of spaces **before and after** each row. More importantly, a common mistake is if you forget the spaces **after** the asterisks and print out the string, it may look correct in your terminal but will fail the autograder.
2. Append the newline character `'\n'` after the last space (if any) in every row **including the last row**. Do not append anything else. Any extra characters will fail the autograder.
3. You must implement this method recursively (which means calling recursiveHourglass within the recursiveHourglass method). We will be testing to see if your answer is correct and if you actually made recursive calls. Hence if you use loops in any way to implement your method, your grade for this portion of the assignment will result in a 0.
4. You must not implement or use any other helper methods to solve this problem and must only rely on `getLine()` and `recursiveHourglass()`.

MUST use it in your implementation below. Though however you use it is up to you.

Part 3: Compile, Run, and UnitTest Your Code (10 points)

In this part of the assignment, you need to implement your own test cases in the method called `unitTests` for both files (`FunWithArrays.java` and `RecursiveHourglass.java`)

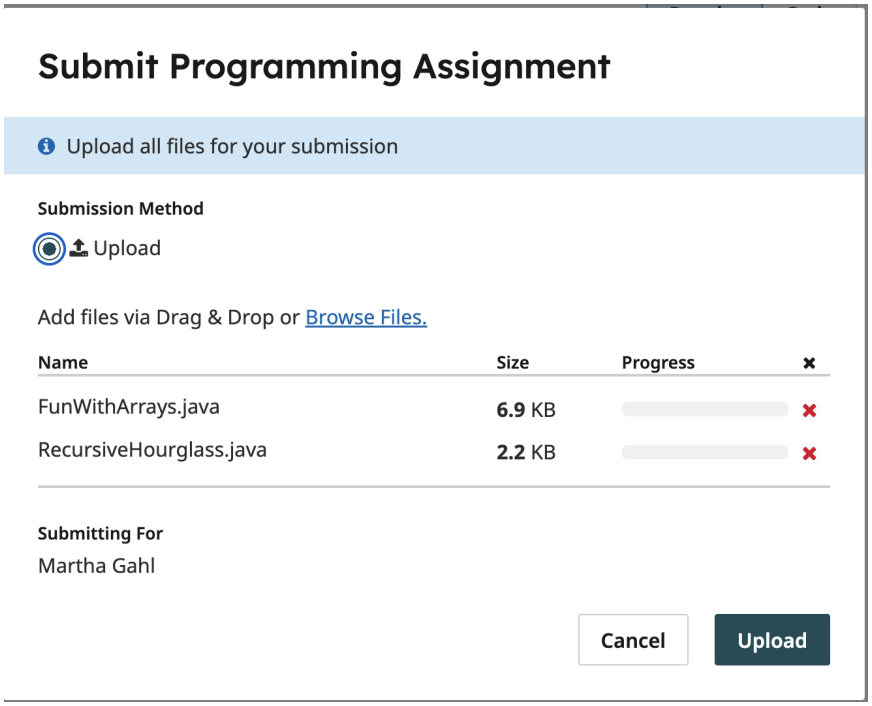
In the starter code, several test cases are already implemented for you. You can regard them as examples to implement other cases. The general approach is to come up with different inputs and manually give the expected output, then call the method with that input and compare the result with expected output.

You are encouraged to create as many test cases as you think to be necessary to cover all the edge cases. The `unitTests` method must `return true` only when all the test cases are passed. Otherwise, it must `return false`. **To get full credit for this section, for each of your eight methods above, you must have at least 3 total test cases that cover different situations (including the ones we have provided).** In other words, you will need to create two more tests for**,** `findPosition`, two more tests for `findAvg`, two more tests for `arrayCopy`, two more tests for `containsIntegerSequence`, two more tests for `reverseArray`, two more tests for `findRange`, two more tests for `arrayAverage`, and two more tests for `recursiveHourglass`.

Submission

You're almost there! Please follow the instructions below carefully and use the **exact submission format**. Because we will use scripts to grade, **you may receive a zero** if you do not follow the same submission format.

1. Open Gradescope and login. Then, select this course → PA4.
2. Click the DRAG & DROP section and directly select the required files `FunWithArrays.java`, and `RecursiveHourglass.java`. Drag & drop is fine. Please make sure you do not submit a zip, just the two files in one Gradescope submission. Make sure the names of the files are correct.
3. You can resubmit unlimited times before the due date. Your score will depend on your final (most recent) submission, even if your former submissions have higher scores.
4. Your submission should look like the below screenshot. If you have any questions, feel free to post on Piazza!


Submit Programming Assignment
Upload all files for your submission
Submission Method
Upload
Add files via Drag & Drop or Browse Files.
Name
Size
Progress
FunWithArrays.java
6.9 KB
RecursiveHourglass.java
2.2 KB
Submitting For
Martha Gahl
Cancel
Upload