**CS102 - Algorithms and Programming II**

**Lab Programming Assignment 5**

**Fall 2021**

| **ATTENTION:**   1. Compress all of the Java program source files (.java) files into a single zip file. 2. The name of the zip file should follow the below convention:   **CS102\_Sec1\_Asgn5\_YourSurname\_YourName.zip**   1. Replace the variables “YourSurname” and “YourName” with your actual surname and name. 2. You may ask questions on Moodle. 3. Complete **all** of the assignment during the lab session and upload the above zip file to Moodle by the deadline. |
| --- |

**The work must be done individually. Codesharing is strictly forbidden. We are using sophisticated tools to check the code similarities. The** [**Honor Code**](https://docs.google.com/document/d/1v_3ltpV_lClLsROXrMbojyuv4KrFQAm1uoz3SdC-7es/edit) **specifies what you can and cannot do. Breaking the rules will result in disciplinary action.**

**Q1 [50 pts.].** In the first part of the lab, you will implement a program that fills a box with a character that is given by the user. A 2D char array will be used for the box. User will give its dimensions, both row and column count. The edges of the box will be a specified character (#) and the rest will be initially empty. Then, the program should fill the box, in other words fill the 2D array, recursively. Please read the notes at the end very carefully in order to understand part 2 clearly.

**FillBox –** One class will be used for this lab. The class that has a tester (main) and other static methods for the first part and second part.

This class should have the following methods:

* char[][] createArray(int row, int column): Creates and returns a 2D character array according to parameters

# Remember to specify edges as “#” and the others (elements inside the box) as space

* void printArray(char[][] array): Prints the array elements , see example output on the next page.
* void fillBox(char[][] array, int row, int column, char ch): Fills inside of the array with the parameter “ch” recursively until the method reaches one of the edges.
  + For this method, there is more than one solution that is recursive.
* void main(String[] args):Get row, column and the character from the user in order to create the array and fill the inside of the box. Create the array by using the “createArray” function. Print the result and call the “fillBox” function and print the result again.

Example output for 10x10 array:

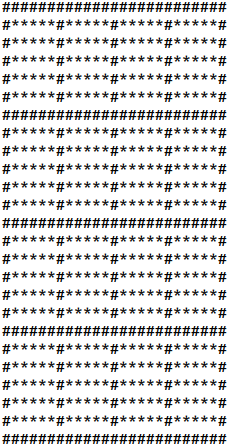




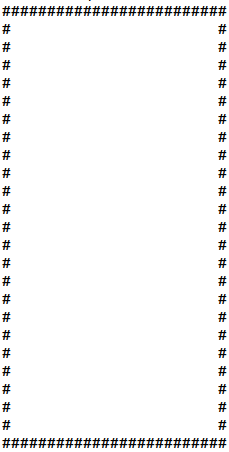
**Q2 [50 pts.].** In this part of the lab, you will add one more recursion method and test this in the tester. For this part, a square matrix should be used.

void drawFigure(char[][] square, int x, int y, int size): Divides the box into 4 pieces in each recursive step. When the piece is too small to divide into 4, this function calls the “fillBox” method in order to fill the inside of the piece and stops.

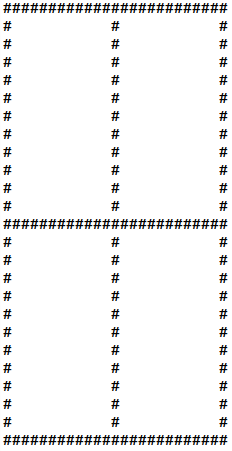
* char[][] square is the 2D matrix that you will paint/fill in.
* int x, int y are respectively the x and y coordinates of the top-left corner of the quadrant
* int size is the width (or height) of the square. (Hint: This should ideally be halved at each recursive step
* In the base case, if the square matrix’s dimensions are less than 7, the method should call the “fillBox” method and stop.
* In the recursion case, the box should be divided into 4 quadrants by replacing spaces with the edge character (“#”). Then, the method should call itself for each of the quadrants (in total 4).
* Assume that at each step the resulting shape is a perfect square. Specifically, please choose the dimension from the following set so that that will be guaranteed: 7, 13, 25, 49, ...., (3 \* 2n + 1). Get dimension from the user and print the result
* Example output for 25x25 array should look like this:

****

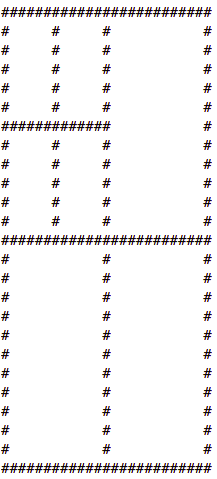
* The below images show the intermediate steps to end up with the final result.

****

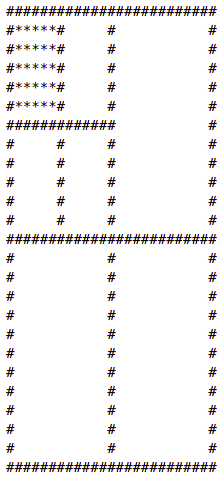
The above image is the initial state of the square matrix when you call createArray method. This matrix will be the initial parameter of the drawFigure method.

****

Before recursing, you will create a plus (+) shape passing through the middle of the square at both axes.

****

Now, you will recurse on 4 of the quadrants. The above image shows the situation when you recurse on the top left quadrant. The top left quadrant did the same and painted a plus (+) shape. It will then recurse on its top-left quadrant.

****

Then, the top-left quadrant of the top-left quadrant has a size of less than 7. So, we hit the base case. Therefore, it immediately calls fillBox which puts the asterisks. The recursion will proceed that way to fill the remaining pieces.

**NOTES**:

* Please reuse the available methods as much as you can instead of repeating the same code in different methods.
* Please comment your code according to the documentation and commenting conventions used in the textbook.