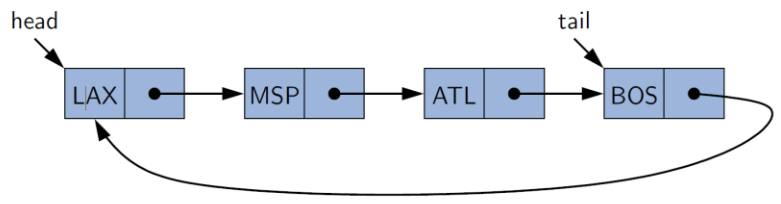
COSC 222: Data Structures

Lab 3 – Queue and Introduction to Trees

Total Marks: 20+2 marks

Question 1: Circular Queue [8 marks]

In our class, we discussed the concept of the circular queue. We can make some small changes to the linked list code for the queue (check lecture 5 slides 10-19) to implement a circular queue. Note that in a circular queue, the `link` (or `next`) reference of the `rear` (or `tail`) node is set to refer back to the `front` (or `head`) of the list (rather than `null`) as you can see in the image below.



Write a program to implement a circular queue using a linked list. You will use the supplied **Node.java** class. Complete the class **LinkedListCircularQueue.java** by writing the following methods:

- `LinkedListCircularQueue()`: Constructor to create an empty queue (1 mark)
- `enqueue()`: Adds the given elements to the rear of the queue (2 mark).
- `dequeue()`: Removes and returns the front element from the queue (2 mark).
- `isEmpty()`: Returns true if this queue is empty; false otherwise (1 mark).
- `first()`: Returns the top element from the queue *without removing it* (1 mark)
- `size()`: Returns the number of elements in the queue (1 mark)

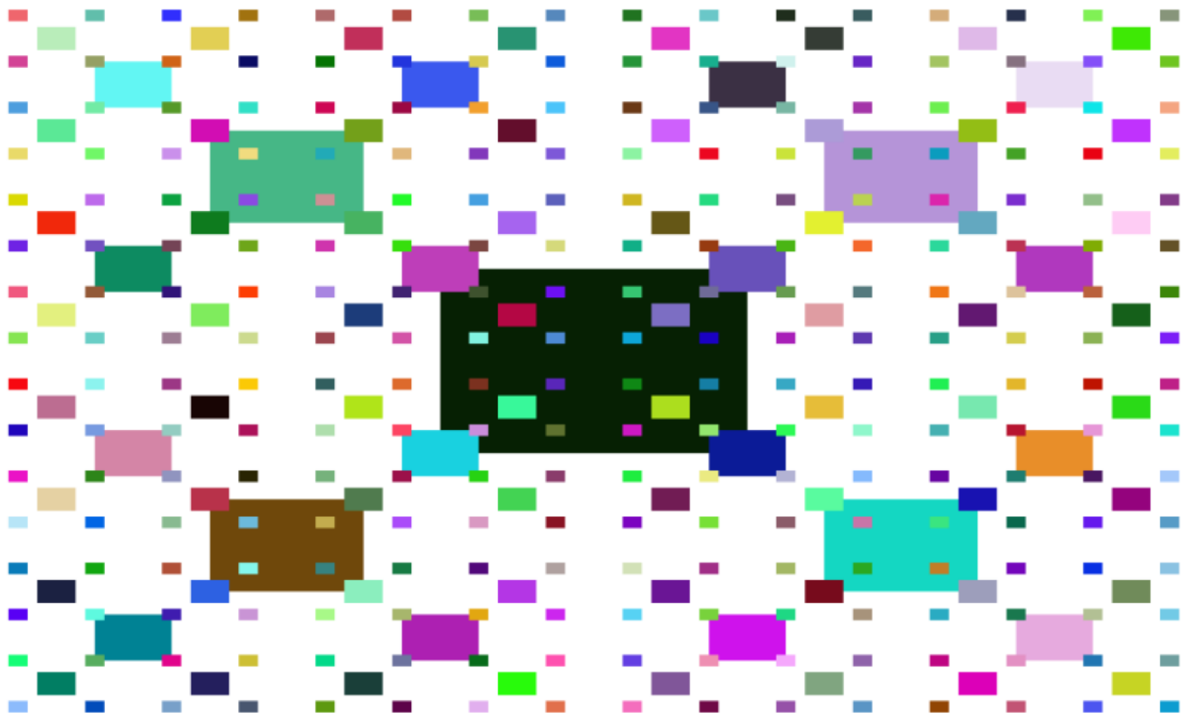
If you complete these methods correctly, you should see the following output when running the **CircularQueueTest.java** file.

Sample Output:

```
Queue size: 2
Queue contents: 1   2

Queue size: 0
Queue contents: 3

Queue is empty, can't remove any item

First item: 4
Queue contents: 4   5
```

Question 2 (Recursion): Recursive Rectangle and Circle [8 + 2 marks]
Begin with the files **RecursiveFill.java** and **StdDraw.java**. *Do not edit **StdDraw.java***.

Part A [8 marks]:
Complete the method `recursiveRectangleFill(double minX, double maxX, double minY, double maxY)` as described below. If you complete the method properly, you will create a randomly coloured, recursively defined image similar to the one shown here.



To fill a rectangular area of the window, `recursiveRectangleFill()` must take the following parameters: `minX`, `maxX`, `minY`, `maxY` (of a window/canvas). Place a rectangle at the center of the window filled with a solid random color. The height and width of the rectangle should be equal to one-fourth of the width and height of the window, respectively. Use `minX`, `maxX`, `minY`, `maxY` parameters to divide the area into four equal pieces (i.e. by dividing the ranges of both the X and Y parameters into halves). These four pieces should again be recursively filled using this same method. (Do not write four copies of almost identical code! Use loops!). Do not draw anything in any rectangular area with a height or a width less than `MIN_SIZE` pixels.

Important Note: the x and y coordinates in `StdDraw` are in the range 0 to 1, but `MIN_SIZE` is in pixels (this will make it work on any size window consistently). The constants `WIDTH` and `HEIGHT` give the size of the entire window (in pixels) and will allow easy conversion to pixels from X or Y values if needed.

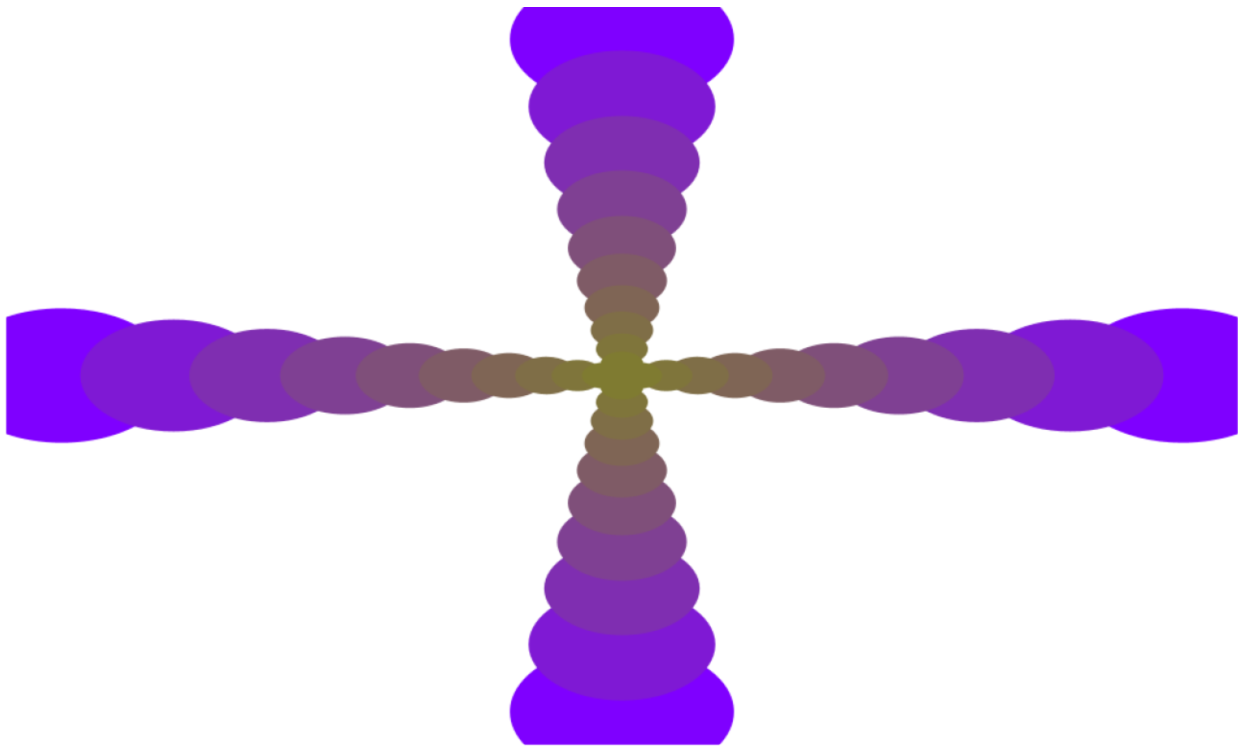
Use StdDraw graphics to draw the graphics. The only commands you need for Part A are:

- `StdDraw.filledRectangle(xc,yc,hw,hh)`: Draw a solid rectangle with a center at (`xc`,`yc`) and the given half width (`hw`) and half height (`hh`).
- `StdDraw.setPenColor(int r, int g, int b)`: Set the colour to have the given amount of red, green, and blue. The values must be integers between 0 and 255. For each int, use `Math.random()` to generate a random double from 0 to 1, multiply that double by 255, and then cast to an int. This will result in a random colour.

Part B [2 BONUS]:
Complete the `recursiveBonus()` method using the following information.

1. Draw circles (not rectangles) [0.5 mark]
2. Color in a defined gradient (not random). Gradient is similar to the image below. (Hint: use `centerX` for red, `centerY` for green and `255*radius*10` for blue) [0.5 mark]
3. Same number and position of shapes as the image below (hint: use `MIN_SIZE_BONUS` instead of `MIN_SIZE`. In each recursive call, change the `radius` by dividing the its value by 1.20. You may call `filledCircle` four times to draw the circles in different directions) [1 mark]

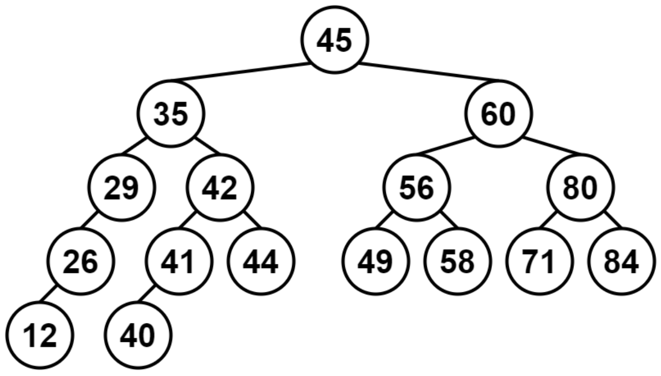
If you edit all aspects of the method properly, you should create a recursively defined image identical to the one shown here.



Question 3 (Tree Basics): Tree Terminology [4 marks]
This question does not require coding. You will need to submit a PDF file with your answers. This can be handwritten and scanned, drawn on a tablet, or created using a diagramming program and word processor. Name your file **Lab3Question3.pdf**.

From the tree given below, write/find/circle/label an example of each of the following terms.

- Subtree
- Height of the tree
- Level of 71
- Descendants of 44
- Leaf nodes
- Depth of 41
- Degree of 80
- Degree of the tree



Submission Instructions:

- Create a folder called "Lab3_<*student_ID* >" where <*student_ID* > is your student number/ID (e.g. **Lab3_12345678**). Only include the mentioned files in the folder. **DO NOT** include any other files (e.g., .class, .java~ or any other files)
  - For Question 1, include your **LinkedListCircularQueue.java** file.
  - For Question 2, include your **RecursiveFill.java** file.
  - For Question 3, include your **Lab3Question3.pdf** file.
- Make a **zip file** of the folder and upload it to Canvas.
- To be eligible to earn full marks, your Java programs **must compile and run** upon download without requiring any modifications.
- These assignments are your chance to learn the material for the exams. **Code your assignments independently.**