```
// DesignMain.cpp
// Author: Jonathan Yu
// Purpose: This program will segment an image using recursion and
//          linked list.


#include "header file for Image object"
#include "header file for Container object"

// main method contains necessary functions
// Preconditions:   GIF file exists and is a valid GIF image
// Postconditions:  Outputs a segmentated version of GIF image and information
//                  about the image to the console
int main() {

    Input GIF image to a Image object
    Construct an output Image object of segmented version of image

    Construct a Container object for merged group, containing all connected groups

    Set a counter for number of segments found

    // Traverses through each row/col of input image to create segmented version
    Set for loop from first row to last row of image {}
        Set for loop from first column to last column of image
            If output image is not colored at given pixel
                Construct a Container object for a connected group
                Add first pixel as the seed pixel of the group
                Call function that creates the connected group
                Call function that modifies the output Image
                Add connected group to merged group
                Increment segment counter

    Output segmented image to a file

    // output information of segmented image to console
    Output total number of segments found, number of pixels in merged group,
    and average color (red, green, blue values) of merged group to console

}

// Create connected group of pixels close to the seed pixel
// Preconditions:   Row and col values are within image boundaries,
//                  input/output Image and Container group objects are passed
// Postconditions:  Connected group of pixels is built
void someRecursiveFunction(input Image object, output Image object,
    connected group, row, col) {

    Set base case to check if given row and col are within bounds of input image
```

```
            Return if false
        Set base case to check if pixel at given row/col of output image is already
          colored
            Return if false
        Create a pixel object of given row and col
        If pixel at given row and col is close enough to seed of connected group
            Add pixel to connected group
            Mark the pixel of given row/col in output image
            Call recursive function with row value incremented by one
            Call recursive function with col value incremented by one
            Call recursive function with row value decremented by one
            Call recursive function with col value decremented by one

    }

    // Modifies the output image by segmenting
    // Preconditions:   Connected group contains at least one pixel and row/col
    //                  values are within image boundaries
    // Postconditions:  Pixels of output image are segmented with given connected
      group
    void someRecursiveFunction2(output Image object, connected group,
        row value, col value) {

        Set base case to check if given row/col values are within image boundaries
            Return if false
        If pixel at given row/col is uniquely marked in output image
            Change pixel to average color of connected group
            Call recursive function with row value incremented by one
            Call recursive function with col value incremented by one
            Call recursive function with row value decremented by one
            Call recursive function with col value decremented by one
    }
```