```
1  // ContainerDesign.cpp
2  // Author: Jonathan Yu
3  // Purpose: This file defines the member functions of Container object.
4
5  Include Container header file
6  Include Math class
7
8  // Constructor
9  // Preconditions:   none
10 // Postconditions:  constructs a Container object
11 Create a Container object constructor
12     Set head pointer to nullptr
13     Set size to zero
14
15 // Copy constructor
16 // Preconditions:   Container object is passed in
17 // Postconditons:   Copy of passed Container object is constructed
18 Create a Container object copy constructor with Container
19 object passed in
20     Set head pointer to nullptr
21     Set size to zero
22     Assign Container to other container
23
24 // Destructor
25 // Preconditions:   none
26 // Postconditions:  all nodes are removed in Container
27 Create a Container object destructor
28     Calls function that removes all nodes
29
30 // Overloader for operator=
31 // Preconditions:   Container object is passed
32 // Postconditions:  Container is assigned to other Container object
33 Create an operator= overloader function with Container object passed in
34     If this and other Container object are same, return this
35     Call function to remove all nodes
36     If head pointer of other Container object points to nullptr
37         Set head pointer to nullptr
38     Else
39         Assign seed pixel to other seed pixel
40         Create a new node and set head pointer to it
41         Assign head pixel data to other head pixel data
42         Create a new pointer for this and other Container object
43         While other pointer does not equal nullptr
44             Create a new Node in this Container object
45             Copy over the image pixel values
46         Set last node in this Container object pointing nullptr
47     Update the size
48     Update the color values of red, green, and blue
49     Return this Container object
```

```
50
51  // Compares two Container objects
52  // Preconditions:   Container object is passed in
53  // Postconditions:  Return true if both Container objects are same, false
      otherwise
54  Create a function that compares two Container objects
55      If both objects do not have any nodes, return true
56      If head pointer of this Container points to a node and
57      head pointer of other Container points to nullptr, or vice versa,
58      return false
59
60      Create a pointer for this Container object
61      Create a pointer for other Container object
62
63      While this or other pointer does not equal nullptr
64          If the color values of the node from this and other
65          Container objects are not equal, return false
66
67      Return true
68
69  // Checks to see if Container object is empty
70  // Preconditions:   Container object is passed in
71  // Postconditions:  Return true if empty, false otherwise
72  Create a function that checks if Container object is empty
73      Return true if size equals zero, false otherwise
74
75  // Adds pixel nodes to Container group
76  // Preconditions:   Passes in a pixel object
77  // Postconditions:  Adds pixel to Container with size and colors updated
78  Create a function that adds pixels to the Container group
79      If head pointer equals nullptr
80          Create a new node
81          Assign node pixel value to passed pixel
82      Else,
83          Traverse through the end of the linked list in this Container
84          Add a new node with the pixel value assigned to pixel passed
85      Call function that updates color values of Container
86      Increment size of Container
87
88  // Merges two Container objects
89  // Preconditions:   Container object is passed in
90  // Postconditions:  Merges Container object with updated size and color values
91  Create a function that merges Container object with another
92      If other Container object contains nodes
93          Create a pointer to traverse linked list of other Container
94          If this Container does not contain any nodes
95              Create the first node
96              Copy node pixel values
97              Assign node to head
```

```
98          Create a pointer to traverse through this linked list
99          While other pointer does not equal nullptr
100             Create a new node for this linked list
101             Copy the pixel values
102         Update total color values of Container object
103         Update the size
104
105 // Creates the seed
106 // Preconditions:   Pixel is passed in
107 // Postconditions:  Seed is updated with pixel values
108 Create a function that adds a pixel object as the seed
109     Assign seed to passed pixel
110
111 // Returns average pixel of Container group
112 // Preconditions:   none
113 // Postconditions:  Returns pixel
114 Create a function that returns pixel of average values
115     Create a new pixel object
116     Assign pixel colors to red, green, and blue average
117     Return pixel
118
119 // Returns average red value of Container
120 // Preconditions:   none
121 // Postconditions:  Returns average red value
122 Create a function that returns average red value of Container
123     Return total red value divided by size
124
125 // Returns average green value of Container
126 // Preconditions:   none
127 // Postconditions:  Returns average green value
128 Create a function that returns average green value of Container
129     Return total green value divided by size
130
131 // Returns average blue value of Container
132 // Preconditions:   none
133 // Postconditions:  Returns average blue value
134 Create a function that returns average blue value of Container
135     Return total blue value divided by size
136
137 // Returns Container size
138 // Preconditions:   none
139 // Postconditions:  Returns number of nodes
140 Create a function that returns the number of nodes in Container
141     Returns the size
142
143 // Checks whether given pixel is close enough to seed pixel
144 // Preconditions:   pixel object is passed in
145 // Postconditions:  Return true if pixel is close to seed pixel, false otherwise
146 Create a function that checks whether a pixel is close enough to the seed
```

```
147      If the total difference between the red, green, blue values of seed and
148      pixel is less than 100, return true, otherwise false
149
150  // Updates the red, green, and blue values of Container
151  // Preconditions:   pixel is passed in
152  // Postconditions:  red, green, and blue values of Container are updated
153  Create a private function that updates the pixel values with passed pixel
154      Adds the passed pixel values of red, green, and blue
155
156  // Updates the red, green, and blue values of Container
157  // Preconditions:   Container object passed is valid
158  // Postconditions:  red, green, and blue values of Container are updated
159  Create a private function that updates the pixel values with passed Container
160      Adds the red, green, and blue values of passed Container
161
162  // Removes the nodes in the Container
163  // Preconditions:   Container is not empty
164  // Postconditions:  Nodes in the Container are removed
165  Create a private function that removes all nodes in Container
166      While Container is not empty
167          Create a temporary pointer pointing to head node
168          Set head pointer pointing to next node
169          Delete temporary pointer
170
171
172
173
174
175
176
```