The PixelList Class

Lab 9 developed the class Pixel that models a RGB pixel. For this lab assignment, you will design and implement a class named PixelList that stores a dynamic list of Pixel objects. The class must have the following constructors, member functions, and overloads.

1.1 Constructors

Implement (at least) the following constructors. Other constructors may be necessary.

- Null constructor - Empty list
- Copy constructor - Makes a deep copy of a PixelList
- Pixel constructor - Create an PixelList from a Pixel.

1.2 Destructor

Properly delete the pixel list.

1.3 Size of the List

The member function size() must return the number of pixels in the list.

1.4 Appending Elements

Implement append(Pixel pixel) method that will append a new Pixel (with value pixel) to the end of the list. For example, the following code segment would create the list \{[8, 8, 8], [18, 18, 18]\}

```cpp
PixelList pList; ///< null constructor
pList.append(Pixel(8, 8, 8));
pList.append(Pixel(18, 18, 18));
```

1.5 Assignment Operator

Make a deep copy of the righthand-side PixelList (or Pixel) and assign it to the lefthand-side PixelList.

1.6 + Operator

Addition of the PixelList can occur with the following data types. All operations must be symmetric and always return a PixelList.

- PixelList - Add corresponding Pixel elements. If one PixelList has fewer elements than the other, assume the missing elements are zero. For example if the PixelList object a stores \{[1, 1, 1], [2, 2, 2]\} and the PixelList object b stores \{[10, 10, 10], [20, 20, 20], [30, 30, 30]\}, then the operation a + b results in \{[11, 11, 11], [22, 22, 22], [30, 30, 30]\}.

- Pixel - As described in the previous operation, adds a Pixel value to the first pixel in the list.

1.7 Logical Operators

Logical operators must be able to compare a PixelList with another PixelList or a Pixel. All operations must be symmetric.

- operator== - Return true if all the corresponding Pixel objects in the lists are equivalent.
- operator!= - Return true if any of the corresponding Pixel objects in the lists are not equivalent.
1.8 Output

The insertion operator \( \ll \) should print out the elements in the list in the following manner. Assume \( a \) is an\nPixelList with values \{[1, 2, 3], [10, 20, 30]\}. If \( a \) is printed to the screen

```cpp
cout << a << '\n';
```

the output must look like

\[
\{[1, 2, 3], [10, 20, 30]\}
\]

2 Programming Points

You must adhere to all of the following points to receive credit for this program.

1. Turn-in (print-outs and electronically) the files for this program.

2. You must submit the following 7 files (use the names listed below).

   - **PixelList** must be broken into 3 files
     - `pixellist.h` Contains the **PixelList** class definition.
     - `pixellist1.cpp` Contains half of the **PixelList** member definitions.
     - `pixellist2.cpp` Contains remaining half of the **PixelList** member definitions.
   - **pixel.h** and **pixel.cpp** Since the **PixelList** class uses the **Pixel** class, include your **Pixel** class files from lab 9 (be certain **Pixel** works correctly).
   - **driver.cpp** A **driver** program that tests the **PixelList** class.
   - **makefile** A makefile to compile the driver program. Note, the makefile must also compile the necessary **Pixel** class files and have a **make clean** option.

3. All arrays must be dynamically allocated with **no wasted space**! Therefore, all arrays must be dynamically sized to store only the information required. Be certain **no** memory leaks occur.

4. Perform appropriate error checking.