CSC 112 Lab 11
The PixelLinkList Class
Spring 2015

Due
3:00pm Friday, 4/24

1 PixelNode Class Description

Design and implement a `class` called `PixelNode` that contains a `Pixel` object (defined in lab 9) and a pointer to the next node. The `PixelNode` must have the following member functions.

- The `PixelNode class` must have the following constructors
  - Null constructor must create an empty `PixelNode`
  - `Pixel` constructor must creates a `PixelNode` from a `Pixel`
  - Copy constructor must make a deep copy of the `PixelNode` argument (copy of the current `PixelNode` and any subsequent`PixelNode` object)
- Destructor must delete the current `PixelNode` and any subsequent `PixelNode` objects.
- Assignment operator must Make a deep copy (copy of the current `PixelNode` and any subsequent `PixelNode` object) of the RHS `PixelNode` and assign it to the LHS `PixelNode`.

2 PixelLinkList Class Description

Design and implement a linked list class called `PixelLinkList`, where the list will be composed of `PixelNode` objects. The `PixelLinkList class` must have the following member functions.

- The `PixelLinkList class` must have the following constructors
  - Null constructor must create an empty `PixelLinkList`.
  - `Pixel` constructor must convert a `Pixel` to a `PixelLinkList`.
  - `PixelList` constructor must converts `PixelList` (defined in lab 10) to a `PixelLinkList`.
  - Copy constructor must make a deep copy of a `PixelLinkList`.
- Destructor must properly delete the `PixelLinkList` object.
- The `append(const Pixel& pix)` member function must append `pix` as the last node of the `PixelLinkList`.
- The member function `size()` must return the number of elements (`PixelNodes`) in the `PixelLinkList`.
- The assignment Operator must make a deep copy of the RHS `PixelLinkList` and assign it to the LHS `PixelLinkList`.
- The insertion operator `<<` should display the elements of the `PixelLinkList`. Print the contents of one `PixelNode` per line. Include the address contained in the `next_` member. For example, consider the following code segment.

```
1 PixelLinkList list(Pixel(10, 10, 10));
2 list.append(Pixel(20, 20, 20));
3 cout << list << '\n';
```

The output **must** look like

```
{[10, 10, 10] (0x856d0f8)}-> {[20, 20, 20] (0)}
```

3 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 all the files necessary to compile and link an executable program that utilized the `PixelLinkList` class and `PixelNode struct`. This includes (but is not limited to) the following files (use the names listed below).
   - `pixellinklist.h` contains the `PixelLinkList` class definition.
   - `pixellinklist1.cpp` contains the `PixelLinkList` member definitions.
   - `pixelnode.h` contains the `PixelNode` class declarations.
   - `pixelnode.cpp` contains the `PixelNode` member definitions.
   - `driver.cpp` is a *driver* program that tests the `PixelLinkList` class.
   - `pixellist.h`, `pixellist1.cpp`, `pixellist2.cpp`, `pixel.h`, and `pixel.cpp`
   - `makefile` is a makefile to compile the driver program. Note, the makefile must also compile all necessary files, be commented, and have a `make clean` option.
3. All `PixelList`s, `PixelNode`s, and `PixelLinkList`s must be dynamically allocated with **no wasted space!** Be certain **no** memory leaks occur.
4. Perform appropriate error checking.