Overloaded Operators

This assignment requires extending the text-based application for working with PPM images. You will now be able to add two images, take their difference, and multiply or divide them by a factor.

The result will be the ability to blend two images, and to change the overall brightness of an image.

Assignment

In this assignment, you will update the ppm_menu program from the previous assignments. All of the previous assignment’s functionality will remain intact.

The new commands required are:

- **read2**: Read file into input image 2.
- **+=**: Set input image 1 by adding in input image 2.
- **-=**: Set input image 1 by subtracting input image 2.
- ***=**: Set input image 1 by multiplying by a number.
- **/=**: Set input image 1 by dividing by a number.
- **+**: Set output image from sum of input image 1 and input image 2.
- **-**: Set output image from difference of input image 1 and input image 2.
- *****: Set output image from input image 1 multiplied by a number.
- **/**: Set output image from input image 1 divided by a number.

Programming Requirements

**Update** ppm_menu

The following methods must be added to the PPM class declaration in PPM.h and implemented in PPM.cpp.

- **bool operator==( const PPM& rhs ) const;** Returns true if *this has the same number of pixels as rhs. Otherwise returns false.
- **bool operator!=( const PPM& rhs ) const;** Returns true if *this has a different number of pixels than rhs. Otherwise returns false.
- **bool operator<( const PPM& rhs ) const;** Returns true if *this has a fewer number of pixels than rhs. Otherwise returns false.
- **bool operator<=( const PPM& rhs ) const;** Returns true if *this has a few number of pixels than rhs or equal number of pixels. Otherwise returns false.
- **bool operator>( const PPM& rhs ) const;** Returns true if *this has a greater number of pixels than rhs. Otherwise returns false.
- **bool operator>=( const PPM& rhs ) const;** Returns true if *this has a greater number of pixels than rhs or equal number of pixels. Otherwise returns false.
- **PPM& operator+=( const PPM& rhs );** Assumes *this and rhs have the same width and height. Adds the channel values from rhs into the channels for *this. If the resulting value is larger than max color value, set to max color value. Returns *this.
- **PPM& operator-=( const PPM& rhs );** Assumes *this and rhs have the same width and height. Subtracts the channel values from rhs from the channels for *this. If the resulting value is less than 0, set to 0. Returns *this.
- **PPM& operator*( const double& rhs );** Multiplies every channel value of *this by rhs. If the resulting value is larger than max color value, set to max color value. If the resulting value is less than 0, set to 0. Returns *this.
- **PPM& operator/=( const double& rhs );** Divides every channel value of *this by rhs. If the resulting value is larger than max color value, set to max color value. If the resulting value is less than 0, set to 0. Returns *this.
- **PPM operator+( const PPM& rhs ) const;** Creates a new PPM object with the same meta data (height, width, max color value) as *this. Sets the channel values in the new object to the sum of the channel values for *this and rhs. If the value is greater than max color value, set to max color value. Returns the new object.
- **PPM operator-( const PPM& rhs ) const;** Creates a new PPM object with the same meta data (height, width, max color value) as *this. Sets the channel values in the new object to the difference of the channel values for *this and rhs. If the value is less than 0, set to 0. Returns the new object.
- **PPM operator*( const double& rhs ) const;** Creates a new PPM object with the same meta data (height, width, max color value) as *this. Sets the channel values in the new object to the product of the
channel values for \( \star this \) and the value of \( rhs \). If the value is greater than max color value, set to max color value. If the value is less than 0, set to 0. Returns the new object.

- \( \text{PPM operator/}\left(\text{const double}\&\ \text{rhs}\right)\ \text{const}; \) Creates a new \( \text{PPM} \) object with the same meta data (height, width, max color value) as \( \star this \). Sets the channel values in the new object to the division of the channel values of \( \star this \) and by the value of \( rhs \). If the value is greater than max color value, set to max color value. If the value is less than 0, set to 0. Returns the new object.

**Update**  \[\text{ActionData.h,cpp}\]

The application will now need another input image object to be used as the right hand side operand when applying binary operators. Add another \( \text{PPM} \) data member, the “input image 2” to the \( \text{ActionData} \) class.

Required methods of the \( \text{ActionData} \) class:

- \( \text{PPM& getInputImage2();} \) Returns the input image 2.

**Update**  \[\text{image_menu.h,cpp}\]

Add the following function declarations to \( \text{image_menu.h} \) and implement them in \( \text{image_menu.cpp} \). The operator calling functions should use input image 1 as the left hand operand. If the right hand operand is a \( \text{PPM} \) object, use input image 2. If the right hand operand is a numeric value, use \( \text{getDouble} \) to ask the user for the value to use. If the operator does not change the left hand operand, assign the result into the output image.

- \( \text{void readUserImage2( ActionData& action_data );} \) Like \( \text{readUserImage1} \), but stores into input image 2.
- \( \text{void plusEquals( ActionData& action_data );} \) Modifies input image 1 by adding input image 2 to it.
- \( \text{void minusEquals( ActionData& action_data );} \) Modifies input image 1 by subtracting input image 2 from it.
- \( \text{void timesEquals( ActionData& action_data );} \) Modifies input image 1 by multiplying it by the double obtained by calling \( \text{getDouble} \) with a prompt of “Factor? “.
- \( \text{void divideEquals( ActionData& action_data );} \) Modifies input image 1 by dividing it by the double obtained by calling \( \text{getDouble} \) with a prompt of “Factor? “.
- \( \text{void plus( ActionData& action_data );} \) Sets output image to be the sum of input image 1 and input image 2.
- \( \text{void minus( ActionData& action_data );} \) Sets output image to be the difference of input image 1 and input image 2.
- \( \text{void times( ActionData& action_data );} \) Sets output image to input image1 times the double obtained by calling \( \text{getDouble} \) with a prompt of “Factor? “.
- \( \text{void divide( ActionData& action_data );} \) Sets output image to input image 1 divided by the double obtained by calling \( \text{getDouble} \) with a prompt of “Factor? “.

The following functions will require updates to their implementations in \( \text{image_menu.cpp} \).

- \( \text{void configureMenu( MenuData& menu_data );} \) add the new actions with the names and descriptions listed below.

<table>
<thead>
<tr>
<th>Command Name</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>read2</td>
<td>readUserImage2</td>
<td>“Read file into input image 2.”</td>
</tr>
<tr>
<td>“+”</td>
<td>plus</td>
<td>“Set output image from sum of input image 1 and input image 2.”</td>
</tr>
<tr>
<td>“+=”</td>
<td>plusEquals</td>
<td>“Set input image 1 by adding in input image 2.”</td>
</tr>
<tr>
<td>“-”</td>
<td>minus</td>
<td>“Set output image from difference of input image 1 and input image 2.”</td>
</tr>
<tr>
<td>“-=”</td>
<td>minusEquals</td>
<td>“Set input image 1 by subtracting input image 2.”</td>
</tr>
<tr>
<td>“*”</td>
<td>times</td>
<td>“Set output image from input image 1 multiplied by a number.”</td>
</tr>
<tr>
<td>“*=”</td>
<td>timesEquals</td>
<td>“Set input image 1 by multiplying by a number.”</td>
</tr>
<tr>
<td>“/”</td>
<td>divide</td>
<td>“Set output image from input image 1 divided by a number.”</td>
</tr>
<tr>
<td>“/=”</td>
<td>divideEquals</td>
<td>“Set input image 1 by dividing by a number.”</td>
</tr>
</tbody>
</table>

**Update**  \[\text{Makefile}\]

If you do not have a \( \text{clean} \) target, create one. It should remove any .o files and any executable files. It should not remove any source files.

**Build Requirements**
Additional Documentation

- C++ Reference
- Examples from class
- Digital Image Processing on Wikipedia

Sample PPM Images

- Monet’s Lilies
- Van Gogh’s Starry Night
- Monet + Van Gogh
- Monet - Van Gogh
- Monet *= 1.5
- Van Gogh /= 2.0

Show Off Your Work

To receive credit for this assignment, you must

- complete the unit tests available in CodeGrinder
- use git to add, commit and push your solution to your repository for this class.

Additionally, the program must build, run and give correct output.

Extra Challenges (Not Required)

- Create additional operators.