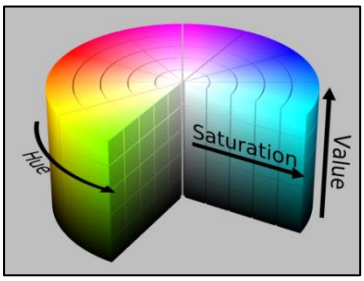
Goal:

Manipulation of BMP image files:
(1) Change the hue of a color image.
(2) Change the saturation of a color image.
(3) Change the value of a color image.

Objective: Developing experience using the C "address-of" operator.

Background: HSV (*hue-saturation-value*) is a representation of color that rearranges the geometry of RGB in an attempt to be more intuitive and perceptually relevant by mapping the values into a cylinder loosely inspired by a traditional color wheel. The angle around the central vertical axis corresponds to ***hue*** and the distance from the axis corresponds to ***saturation***. The height corresponds to ***value***, the system's representation of the perceived luminance in relation to the saturation.



Download: Download and unpack file lab4.zip from Camino.

BMP Library: Use the following library functions instead of GetRed, GetGrn, GetBlu, PutGrn, PutRed, or PutBlu.

```
RGB GetRGB(IMAGE *image, unsigned row, unsigned col) ;
```

Retrieves the RGB color components of a pixel at a specific row and column.

Note: This function replaces the three bmp1 library functions GetRed, GetGrn and GetBlu.

```
void PutRGB(IMAGE *image, unsigned row, unsigned col, RGB *rgb) ;
```

Replaces the RGB color components of a pixel at a specific row and column.

Note: This function replaces the three bmp1 library functions PutRed, PutGrn and PutBlu.

```
HSV RGB2HSV(unsigned red, unsigned grn, unsigned blu) ;
```

Converts RGB component values to the corresponding HSV component values.

```
RGB HSV2RGB(double hue, double sat, double val) ;
```

Converts HSV component values to the corresponding RGB component values.

Assignment: Complete the source code for each the following three functions that are located within the provided main program (lab4.c):

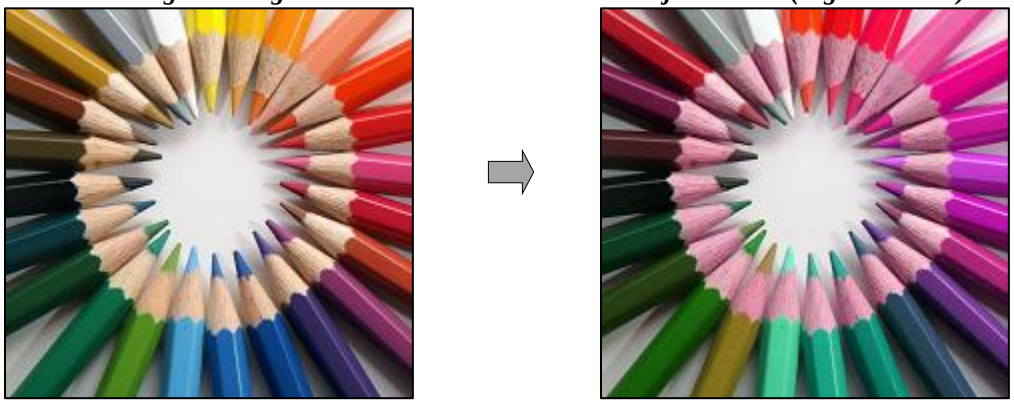
```
IMAGE  *AdjustHue(IMAGE *image, int degrees) ;
```

Adjusts the hue of an image by adding the value of *degrees* to the hue of every pixel and returns a pointer to the image.

The value of parameter *degrees* is limited only by the range of data type `int`.

When done correctly, the result should look similar to the example below:

Original Image ***Adjusted Hue (degrees = -45°)***



```
IMAGE  *AdjustSaturation(IMAGE * image, int percent) ;
```

Adjusts the saturation of all pixels in an image by *percent* and returns a pointer to the image.

Note: The saturation of each pixel is adjusted by the value of parameter *percent*, which must be positive. Saturation is decreased when *percent* is < 100 and increased when it is > 100.

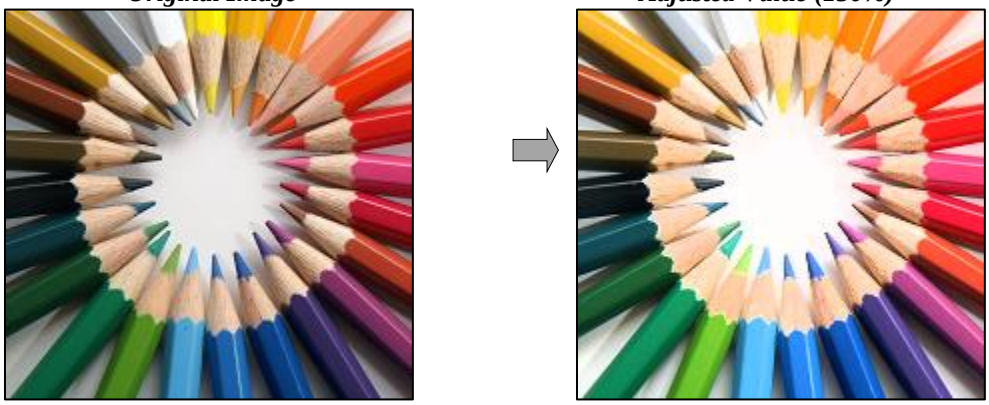
When done correctly, the result should look similar to the example below:

Original Image ***Adjusted Saturation (50%)***

`IMAGE *AdjustValue(IMAGE * image, int percent) ;`
Adjusts the value (perceived luminance) of all pixels in an image by *percent* and returns a pointer to the image. Note: The value of each pixel is adjusted by the value of parameter *percent*, which must be positive. Luminance is decreased when *percent* is < 100 and increased when it is > 100.

When done correctly, the result should look similar to the example below:

Original Image ***Adjusted Value (150%)***



Compilation: Compile and link your program using the following command line:

gcc –o lab4 lab4.c -L. -lbmp

Execution: Execute your program using the following command syntax:

./lab4 *src-file* *dst-file* *{option#}*

When Done: Demonstrate proper operation of your program to the teaching assistant and upload the completed source code for file lab4.c to the lab drop box on Camino. Do not upload any other files.