Question 1. In our representation of images,
- each **Picture** consists of a two-dimensional array of **Pixels**,
- each **Pixel** consists of (among other things) its **Color**.
- each **Color** consists of several components.
Write 1- to 2-sentence answers to each of the below questions about our representation of colors.

a. (5 points) How do we represent/encode color? **RGB value**
   - 8-bit red value
   - 8-bit green value
   - 8-bit blue value

b. (5 points) What range of integer values are available for each color component? $0_{10}$ to $255_{10}$

c. (5 points) Referring to your answer for part b - explain this range of values.
   - 8-bit binary values: $00000000_2$ to $11111111_2$
   - $0_{10}$ to $255_{10}$

d. (5 points) What is luminance?
   - It is the average of three color components (RGB) which translates to the "darkness"/"lightness" of a color.

e. (5 points) What makes a color grayscale?
   - **All RGB values are equal.**
   - e.g. $(128, 128, 128)$

f. (5 points) How many different shades of gray are there? (Justify your answer)
   - **By above definition** $(0,0,0)$ to $(255,255,255)$
   - or $256$. Since $(0,0,0)$ is black and $(255,255,255)$ is white, you could say 254 shades of gray.
Question 2. What \texttt{range} function calls would generate each of the following sequences?

a) (2 points) 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 \hspace{1cm} \texttt{range(16)} \text{ or } \texttt{range(0,16)}

b) (3 points) 0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30 \hspace{1cm} \texttt{range(0,31,3)}

c) (3 points) 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0 \hspace{1cm} \texttt{range(10,1,-1)}

Question 3. When we process pictures we can use one of two approaches:

- a single loop with code like:
  \begin{verbatim}
  for pixel in getPixels(picture):
    # code to process pixel
  \end{verbatim}

- nested loops with code like:
  \begin{verbatim}
  for x in range(....):
    for y in range(....):
      # code to process pixel at x, y coordinate
  \end{verbatim}

(10 points) Briefly describe (1 or 2 sentences) how we know when to use one approach over the other.

The single loop can be used if you want to process every pixel the same regardless of its $x,y$ location.

The nested loops can be used if we only want to loop over a partial picture or if we want to process a pixel based on its $x,y$ location.

Question 4. The following questions refer to the following block of code:

\begin{verbatim}
def problemFour(picture):
    for x in range(0,getWidth(picture)):
        for y in range(0,getHeight(picture)):
            pixel = getPixel(picture,x,y)
            r = getRed(pixel)
            setRed(pixel,r + 40)
\end{verbatim}

a. (5 points) What effect would this function have on the picture’s appearance?

\textit{It would tint the picture redder.}

b. (2 points) Does the function process pixels row-wise or column-wise?
\textbf{Column-wise}

c. (5 points) What would you do if you wanted to reverse this so it processed pixels in the other “direction”?

\begin{verbatim}
switch the two loops: for y:
    for x:
\end{verbatim}

Question 5. (10 points) Draw the output of the following code in the blank “picture” that’s 300 wide by 300 tall:

\begin{verbatim}
picture = makeEmptyPicture(300, 300, white)
addLine(picture, 0, 0, 150, 150)
addRect(picture, 100, 100, 100,100)
addOval(picture, 150, 0, 50, 150)
\end{verbatim}
Question 6. Consider the following piece of code:

```python
def problemSix(picture):
    width=getWidth(picture)
    height=getHeight(picture)
    output = makeEmptyPicture(width, height)
    for x in range(0, width):
        for y in range(0, height):
            source=getPixel(picture, x, y)
            target=getPixel(output, x, y)
            r=getRed(source)
            g=getGreen(source)
            b=getBlue(source)
            if r>=g and r>=b:
                setColor(target, makeColor(255, 0, 0))
            if g>=r and g>=b:
                setColor(target, makeColor(0, 255, 0))
            if b>=r and b>=g:
                setColor(target, makeColor(0, 0, 255))
    return output
```

a. (10 points) Briefly explain what the output picture looks like.

The picture will be posterized to the "true" red, green, or blue colors based on each pixels large RGB component value.

b. (5 points) What would be an appropriate name for this method?

`trueRGBPosterization`
Question 7. (15 points) Write a function called `greenToExtreme(picture)` which takes a picture as a parameter and changes the value of green for each pixel in the original picture to either 0 or 255, whichever is closest to the original green value of the pixel. Red and blue should remain unchanged. This method should not return or display anything on its own.

```python
def greenToExtreme(picture):
    for pixel in getPixels(picture):
        r = getRed(pixel)
        g = getGreen(pixel)
        b = getBlue(pixel)
        if g < 128:
            setColor(pixel, makeColor(r, 0, b))
        else:
            setColor(pixel, makeColor(r, 255, b))
```

```python
def greenToExtreme(picture):
    for x in range(getWidth(picture)):
        for y in range(getHeight(picture)):
            pixel = getPixel(picture, x, y)
            if getGreen(pixel) < 128:
                setGreen(pixel, 0)
            else:
                setGreen(pixel, 255)
```