Section1 - Assignment 1 - Image Scaling and Seam carving: 10% + (2% extra)

Student:

Naphat	Khajohn-udomrith	6188029
Mangkhales	Ngamjaruskotchakorn	6188055
Teekawin	Kirdsaeng	6188077

A1-1 Nearest Neighbor Scaling (1.5%)

Code:

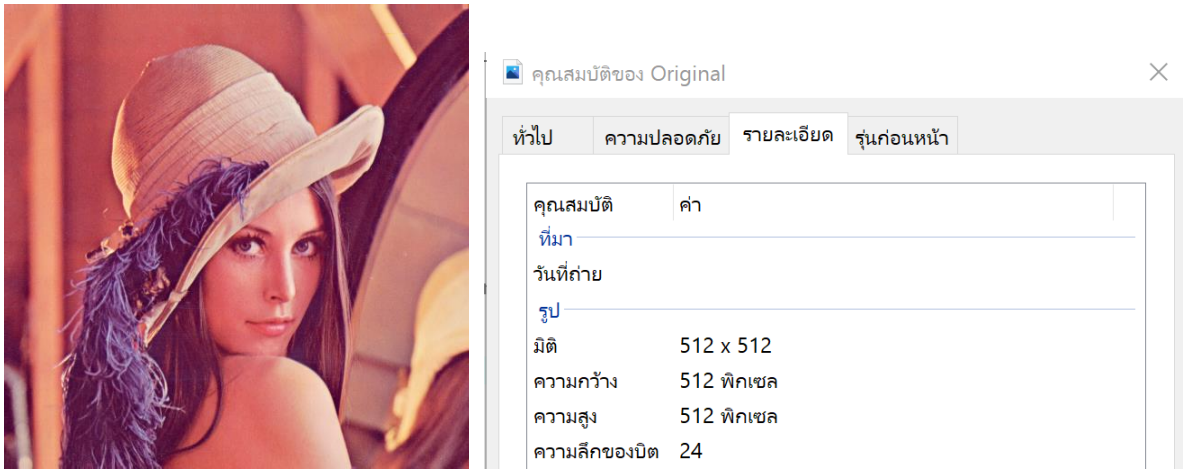
```
// A1-1 Nearest Neighbour Scaling ==========================================================================
void ResizeNearestNeighbor(int newWidth, int newHeight)
{
  println("ResizeNearestNeighbor(param1, param2): param1 = " + newWidth + " param2 = " + newHeight);
  if (originalImage != null)
  {

    if (newWidth != 0 && newHeight != 0)
    {
      UpdateBufferWithOriginalImg();
      bufferImage.resize(newWidth, newHeight);
    }
    else
    {
      if(newWidth==0)
      {
        UpdateBufferWithOriginalImg();
        bufferImage.resize(newHeight, newHeight);
      }
      else
      {
        UpdateBufferWithOriginalImg();
        bufferImage.resize(newWidth, newWidth);

      }
    }
  }
  else
  {
    println("Error: originalImage is null");
  }
}
```

This task is about how to resize the image by first, we will check the original image if it does not exist, it will show the context “Error: originalImage is null”. Next step, we will check the width(newWidth) and height(newHeight). If both of them is not zero, it will update the buffer image with original image by call method “UpdateBufferWithOriginalImg()” and then resize the buffer image to scaling of size that we sent in parameter. But if each of them is zero, it will have to check which one is zero and change the size from zero to the same size with another one such as “width=321 and height=0 it will change the height to the same with width that is 321 then finally, width=321 and height=321”

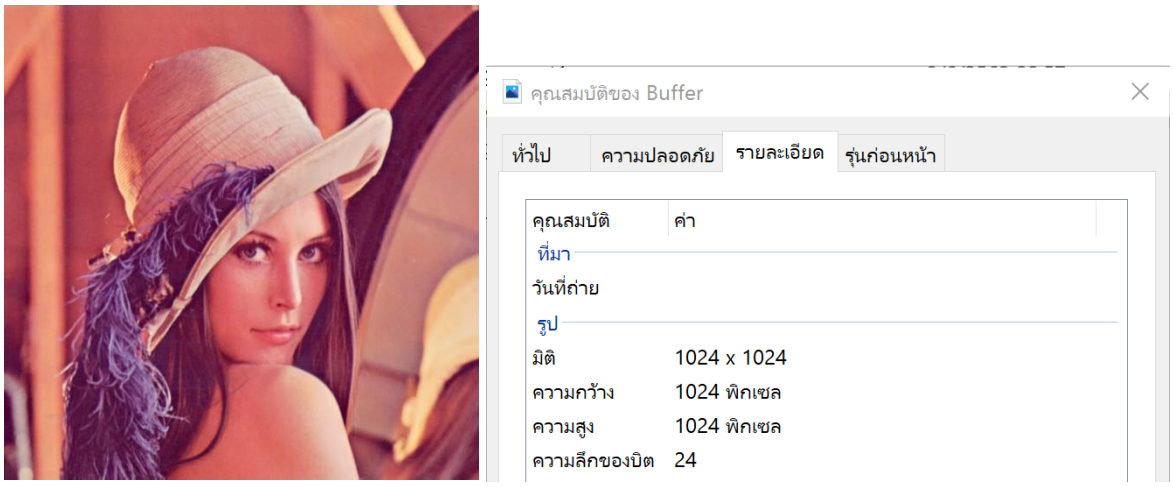
```
// Nearest Neighbour Scaling
imageLib.ResizeNearestNeighbor(1024, 1024);
//imageLib.ResizeNearestNeighbor(256, 256);
//imageLib.ResizeNearestNeighbor(256, 128);
//imageLib.ResizeNearestNeighbor(321, 0);
```

Example of choice that we use is 1024,1024

Output:



Original.png



Buffer.png

From the program, it will have two result that are Original.png and Buffer.png. If you can see the original image have the scaling 512x512 pixel after we use the program it will have the result that is buffer image. From the choice that we use to resize the image to 1024x1024 so the buffer image wil have scaling 1024x1024.

A1-2 Bilinear Interpolation Scaling (1.5%)

Code:

```
// A1-2 Bilinear Interpolation Scaling ================================================================
void ResizeBilinearInterpolation(int newWidth, int newHeight)
{
  int Width = newWidth;
  int Height = newHeight;
  println("ResizeBilinearInterpolation(param1, param2): param1 = " + newWidth + " param2 = " + newHeight);
  if (originalImage != null)
  {

    if (newWidth != 0 && newHeight != 0)
    {
      UpdateBufferWithOriginalImg();
      newWidth = (int) (bufferImage.width*1.6f);
      newHeight = (int) (bufferImage.height*1.6f);
      //PImage newImage = createImage(newWidth, newHeight, RGB);
      for (int x = 0; x < newWidth; ++x) {
        for (int y = 0; y < newHeight; ++y) {
          float gx = ((float) x) / newWidth * (bufferImage.width - 1);
              float gy = ((float) y) / newHeight * (bufferImage.height - 1);
              int gxi = (int) gx;
              int gyi = (int) gy;
              int rgb = 0;
              int c00 = bufferImage.get(gxi, gyi);
              int c10 = bufferImage.get(gxi + 1, gyi);
              int c01 = bufferImage.get(gxi, gyi + 1);
              int c11 = bufferImage.get(gxi + 1, gyi + 1);
              for (int i = 0; i <= 2; ++i) {
                  float b00 = get(c00, i);
                  float b10 = get(c10, i);
                  float b01 = get(c01, i);
                  float b11 = get(c11, i);
                  int ble = ((int) blerp(b00, b10, b01, b11, gx - gxi, gy - gyi)) << (8 * i);
                  rgb = rgb | ble;
            }
            bufferImage.set(x, y, rgb);
        }
      }
      bufferImage.resize(Width, Height);
    }
```

//It is example of the code. (it has more, I can't take it all to the report, but you can see it all in code //file kub Ajarn d(^o^))

It is the same with the A1-1, it uses the same concept but it has to calculate more in each pixel. It will used by take the loop to run in every pixel of image. First, we create the syntax to calculate follow the formula by create method to help:

```
// Note : Two funtion make to help to calculate the question A1-2
//=============================================================================================
  int get(int self, int n) {
        return (self >> (n * 8)) & 0xFF;
  }
  float lerp(float s, float e, float t) {
        return s + (e - s) * t;
  }

  float blerp(final Float c00, float c10, float c01, float c11, float tx, float ty) {
        return lerp(lerp(c00, c10, tx), lerp(c01, c11, tx), ty);
  }
//=============================================================================================
```

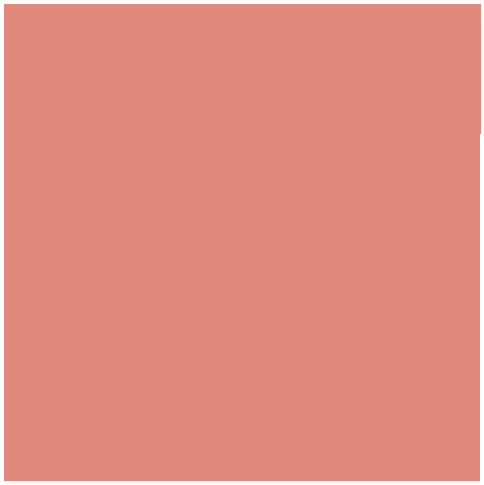
```
// ResizeBilinearInterpolation
//imageLib.ResizeBilinearInterpolation(1024, 1024);
imageLib.ResizeBilinearInterpolation(256, 256);
//imageLib.ResizeBilinearInterpolation(256, 128);
//imageLib.ResizeBilinearInterpolation(321, 0);
```

Example of choice that we use is 256,256

Output:

คุณสมบัติ	ค่า
ที่มา	
วันที่ถ่าย	
รูป	
มิติ	512 x 512
ความกว้าง	512 พิกเซล
ความสูง	512 พิกเซล
ความลึกของบิต	24

Original.png



คุณสมบัติ	ค่า
ที่มา	
วันที่ถ่าย	
รูป	
มิติ	256 x 256
ความกว้าง	256 พิกเซล
ความสูง	256 พิกเซล
ความลึกของบิต	24

Buffer.png

I am not sure that the answer is correct or not but it has the output that already change the scaling to 256x256 pixel

A1-3 Seam Carving (4%)

Code:

```
// A1-3 SeamCarving Scaling ========================================================================
void ResizeSeamCarvingBasic(int newWidth, int newHeight)
{
  println("ResizeSeamCarving(param1, param2): param1 = " + newWidth + " param2 = " + newHeight);
  //TODO-0: calculated how many seam need to be removed from the original image in both vertical

  //TODO-1: resize on the horizontal
  // TODO-1-1: calculated 'vertical seam' to be removed
  Seam verticalseam = GetSingleVerticalSeam();
  // TODO-1-2: Remove the seam then Repeat the process for 1-1
  RemoveSingleSeam(verticalseam);
  ResizeNearestNeighbor(newWidth,newHeight);
  //TODO-2: resize on the vertical
  // TODO-2-1: calculated 'horizontal seam' to be removed
  Seam horizontalseam = GetSingleHorizontalSeam();
  // TODO-1-2: Remove the seam then Repeat the process for 1-2
  RemoveSingleSeam(horizontalseam);
  ResizeBilinearInterpolation(newWidth,newHeight);
}
```

This task wants to calculated the seam to be removed and remove the seam then repeat the process to resize the image. First, we create new seam named “verticalseam” to get the result of method “GetSingleVerticalSeam()” to calculated 'vertical seam' to be removed and then we will remove the seam by calling “RemoveSingleSeam()” and take the seam that we created to the parameter, it will look like this “RemoveSingleSeam(verticalseam)” and then repeat the process for A1-1 to resize like this “ResizeNearestNeighbor(newWidth,newHeight)”. Second, we create new seam named “horizontalseam” to get the result of method “GetSingleHorizontalSeam()” to calculated 'horizontal seam' to be removed and then we will remove the seam by calling “RemoveSingleSeam()” and take the seam that we created to the parameter, it will look like this “RemoveSingleSeam(horizontalseam)” and then repeat the process for A1-2 to resize like this “ResizeBilinearInterpolation(newWidth,newHeight)”.

```
void RemoveSingleSeam(Seam seam)
{
  if (bufferImage != null && seam != null)
  {
    // TODO: Use seam array of marked 1D pixel location to remove data
    // from the bufferImage
    if (bufferImage.width <= 1 || bufferImage.height <= 1)
    {
      throw new IllegalArgumentException();
    }
    for (int x : seam.pixelIndices) {
          if (x < 0 || x >= seam.pixelIndices.length){
            throw new IllegalArgumentException();
          }
    }
    for (int r = 0; r < seam.pixelIndices.length; r++) {
          if (seam.pixelIndices[r] < seam.pixelIndices.length - 1) {
              System.arraycopy(bufferImage.pixels[r], seam.pixelIndices[r] + 1, bufferImage.pixels[r], se
          }
      }

  } else {
    println("Error: bufferImage or seam is null");
  }
}
```

This is the method remove single seam. The process is to remove the seam by first we check the buffer image and seam. If it does not exist it will show “Error: bufferImage or seam is null”. Next step is use to check the argument. Finally, we create loop to copy the data in array.

```
// ResizeSeamCarvingBasic
//imageLib.ResizeSeamCarvingBasic(1024, 1024);
//imageLib.ResizeSeamCarvingBasic(256, 256);
imageLib.ResizeSeamCarvingBasic(256, 128);
//imageLib.ResizeSeamCarvingBasic(321, 0);
```

Example of choice that we use is 256,128

Output:

I don’t have the output, I am only adapt the logic to the code and I can’t finish it (TToTT).