

Department of Computer Science, The University of Hong Kong

**COMP7503
Multimedia Technologies**

Programming Assignment

Deadline: 1 Dec 2018, before 11:55pm

Preamble:

This assignment is composed of two parts: a) Programming Part, and b) Written Part. The purpose of this assignment is to get you familiar with the concepts of 1) Image Quantization; and 2) Lossless Image Compression, through writing computer programs to process some real images. A program template based on Microsoft Visual Studio 2012 development environment will be provided. User interface has been developed so that your focus can be put only on algorithmic aspects.

The programming part requires you to implement an image enhancement application presented in the paper, which enhance subjective image quality for visualization purpose. The written part, on the other hand, requires you to write a report, in which you explain in details your implemented algorithms in the programming part.




Please note that this is a group programming assignment. We expect two students working together to complete this assignment.

Overview:

A program template is first provided to you. Based on this template, you should implement a matched pair of *quantization* and *dequantization* algorithm, and also a matched pair of *compression* and *decompression* algorithm.

Within the template, you should regard a color image as arrays of pixels. Each pixel is being represented by 3 color components, namely Red, Green and Blue components. Each pixel is represented in 24 bits, with 8 bits for each color component.

On receiving the program template, you should be able to compile the program under Visual Studio 2012 without writing any code. The program template will take care of all the issues related to user interface, windows redraw and file read/write.

When you run the program, you can load image files in BMP, JPG, or PNG (without alpha channel) formats through the menu ("File->Open") or by pressing the button . In this assignment, you will be implementing algorithms for quantisation and compression that can be triggered by the press of  and  buttons.

Programming Part:

For the programming part, you only need to focus on four files, namely:

1. *AppQuantize.h* (under VPT/Header Files/App)
2. *AppQuantize.cpp* (under VPT/Source Files/App)
3. *AppCompress.h* (under VPT/Header Files/App)
4. *AppCompress.cpp* (under VPT/Source Files/App)

You may want to have a look at the following functions in the AppQuantize.cpp first.

unsigned char *CAppQuantize::Quantize555(int &qDataSize)

void CAppQuantize::Dequantize555(unsigned char *quantizedImageData, unsigned char *unquantizedImageData)

The *Quantize555* function gives you an example implementation of quantizing 8-8-8 color pixel to 5-5-5 color pixel format. The *Dequantize555* function, on the other hand, gives you an example implementation of dequantizing 5-5-5 color pixel format back to 8-8-8 format.

Upon closer inspection in these example functions, you can see that we have some predefined variables:




1. **int width, height ;**

These two integers would give you the width and height of a given loaded image.

2. **unsigned char *pInput ;**

This is the pointer to the loaded image. The R, G, B component (in 8-bit format) of a pixel at the location (i, j) can be addressed from this pointer as follows

B: **pInput[(i + j * width) * 3 + 0] ;**
G: **pInput[(i + j * width) * 3 + 1] ;**
R: **pInput[(i + j * width) * 3 + 2] ;**

As discussed before, you should be able to compile the program under Visual Studio 2012 without writing any code. You'll then able to run the program by pressing F5 key in Visual studio. You can then load some sample image files provided in the template through the menu ("File->Open") or by pressing the button . After successful loading the image, you can press the button  to bring up the console LOG window for seeing custom messages printed from the program. To test the Quantize555 implementation, you can press the button , and two new windows "5-5-5 Output" and "5-6-5 Output" should popup. You'll be able to see the 5-5-5 quantized version of the loaded image, as well as a black image in the "5-6-5 Output" window as shown in Figure 1 below.

You'll also see in the "Console Log" window some text messages, to indicate the original data size of the 8-8-8 bit image, as well as the size of the quantised version of the image and the resultant compression ratio.

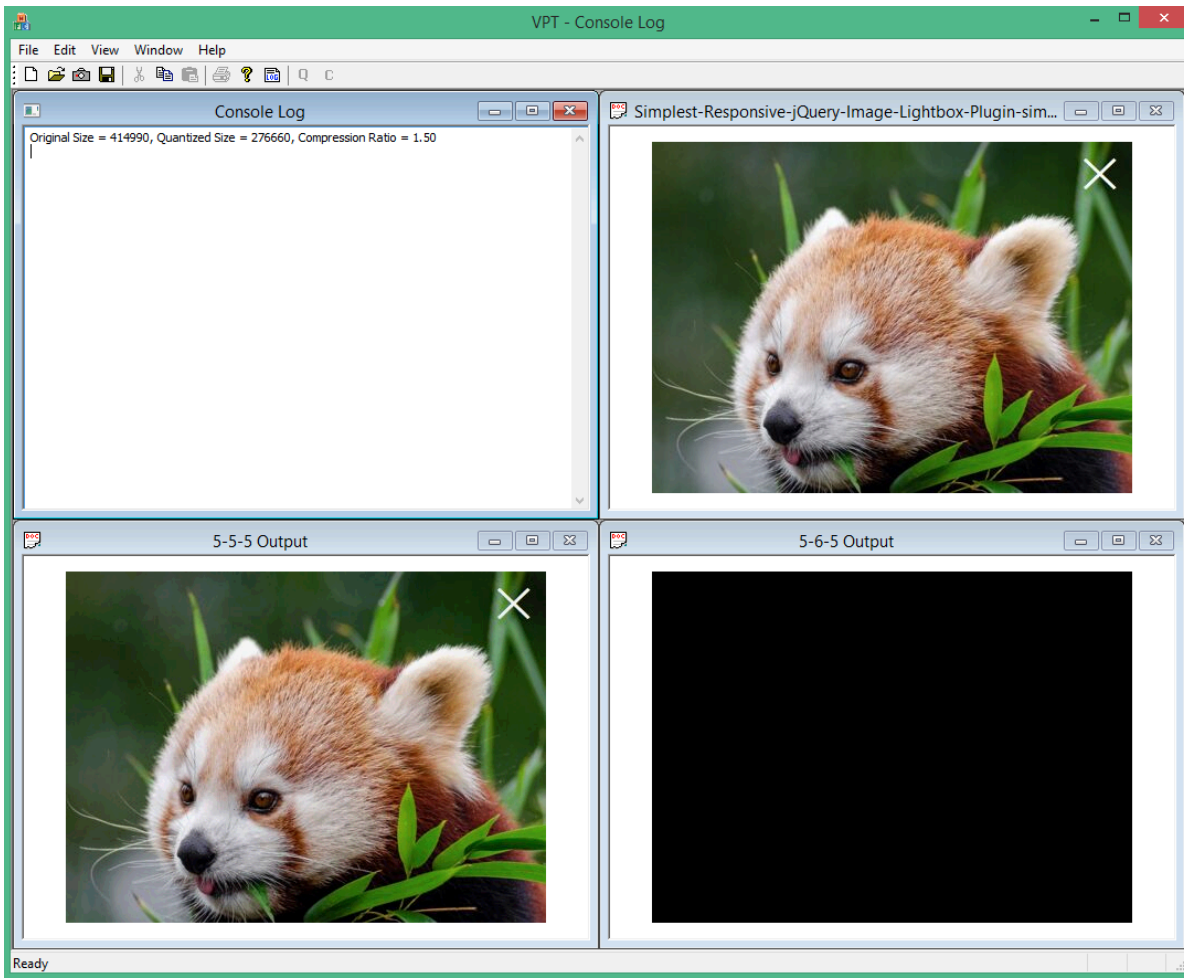


Figure 1: Sample Processing Results

First Programming Task

Your first task of this assignment is to complete the implementation of the following functions in the “*AppQuantize.cpp*” file to realise the conversion between 8-8-8 image and 5-6-5 image formats.

```
unsigned char *CAppQuantize::Quantize565(int &qDataSize) ;
```

```
void CAppQuantize::Dequantize565(unsigned char *quantizedImageData,  
unsigned char *unquantizedImageData) ;
```


Upon successful completion of these functions, you should be able to see meaningful images in the “5-6-5 Output” windows.

Second Programming Task

Your second task of this assignment is to complete the implementation of the following functions in the “*AppCompress.cpp*” file to realise the a lossless compression and decompression algorithms.

```
unsigned char *CAppCompress::Compress(int &cDataSize) ;
```

```
void CAppCompress::Decompress(unsigned char *compressedData, int  
cDataSize, unsigned char *uncompressedData) ;
```

To test the your compression and decompression implementation, you can press the button , and the window “Lossless Decompressed Image” will pop up and you can verify the correctness of your implementation. You may also see in the “Console Log” window for the performance of your compression algorithms, and it shall also alert you if your decoded image is not identical to the original image.

Programming Task Remarks:

Please note that you should keep the following function definition and parameter list unchanged.

```
unsigned char *CAppQuantize::Quantize565(int &qDataSize) ;
```

```
void CAppQuantize::Dequantize565(unsigned char *quantizedImageData,  
unsigned char *unquantizedImageData) ;
```

```
unsigned char *CAppCompress::Compress(int &cDataSize) ;
```

```
void CAppCompress::Decompress(unsigned char *compressedData, int  
cDataSize, unsigned char *uncompressedData) ;
```

Apart from these functions, you cannot do modification to any other code anywhere else. However, you may define helper functions of your own for your algorithms implementations, but these should be isolated functions that can only be referenced or called within the four functions listed above. Any helper functions defined must be contained in the following files only:

1. *AppQuantize.h* (under VPT/Header Files/App)
2. *AppQuantize.cpp* (under VPT/Source Files/App)
3. *AppCompress.h* (under VPT/Header Files/App)
4. *AppCompress.cpp* (under VPT/Source Files/App)

Written Part:

You are required to write a report and describe in details the algorithms you have implemented in the programming part.

Besides, you’ll also need to do the following experiment.

Experiment procedure:

1. Load an image ;
2. Do a compression on the image, and note the compression ratio ;
3. Do a quantization on the same image ;
4. Do a compression on the quantized image (on the 5-5-5 version), and note the compression ratio.

In your report, you should compare the compression ratios obtained in step 2 and step 4, and explain the difference between the two compression ratios.

Assignment Submission:

Assignment must be submitted through the moodle course web. The following files have to be submitted as a one single zip file:

1. *AppQuantize.h*
2. *AppQuantize.cpp*
3. *AppCompress.h*
4. *AppCompress.cpp*
5. *Your written report*

Important Notes:

1. Late submission is subjected to the penalty of 5% deduction per day.
2. Submission late for more than one week will not be entertained.
3. Programs that cannot be compiled will NOT be graded.
4. Plagiarism will be heavily penalized.

=== End ===