

# Assignment #2 Instructions: BMP Image Processing

Through this programming assignment, the students will learn to do the following:

1. Learn to read and write binary files.
2. Gain more experience with using pointers.
3. Gain more experience with dynamic memory allocation and deallocation.

In this assignment, you are asked to manipulate an image from a 24-bit uncompressed bmp file. (The format of bmp files is given in [http://en.wikipedia.org/wiki/BMP\\_file\\_format](http://en.wikipedia.org/wiki/BMP_file_format)). To help with reading and writing the image file, you are given library functions and an example that deal with the bmp file (in a zip file). In the example, the image is flipped horizontally. You are asked to implement some additional functions including the `verticalflip()`, `enlarge()` and `rotate()` functions: the first flips the image vertically, the second is used to enlarge the image by an integer scale factor of 2 or 3; the last one is used to rotate the image either clockwise by 90 degrees or counter-clockwise by 90 degrees.

The program should take the follow command-line options:

```
% bmptool [-f | -r | -l | -s scale| -v ] [-o output_file]
[input_file]
```

Where `-s` means to scale the image by a scale factor of 2 or 3 as indicated on the command line with a 2 or a 3, `-r` means to rotate the image by 90 degrees clockwise, `-l` means to rotate the image by 90 degrees counter-clockwise, `-v` means to flip the image vertically and `-f` means to flip the image horizontally. You can assume for each type (`-s`, `-r`, `-l`, `-f`, or `-v`), the command line has at most one option, but that should not affect your code. Only implement each one one time no matter how many times it is

included. However, the user may present a combination, say, 'bmptool -r -s -l -f -v'. If multiple option types are present, the order for processing the image is that you do scale first, then any rotate options, and then flip vertically and finally flip horizontally.

You are required to use getopt() to process the command-line. If '-o output\_file' is missing, use standard output. If 'input\_file' is missing, use standard input. The options to use standard input or standard output will only be used when chaining commands. Make sure the program returns 0 on success. In that case, one can 'chain' the commands using pipes, like:

```
% bmptool -s 2 1.bmp | bmptool -r | bmptool -f -o 2.bmp
```

Your program needs to provide necessary sanity-check for command line arguments and handle various error conditions and prompt the user with helpful information. You need to use getopt() to process the command line arguments. Test your program with various combinations to make sure it works as expected. You must use dynamic memory to store the content of the new image before writing out to file. You need to reclaim memory afterwards to prevent memory leaks. Please submit your work through Canvas as one zip file called FirstnameLastnameA2.zip. Follow the instructions below carefully (to avoid unnecessary loss of grade). Include your source code, your Makefile, the bmlib files that I have provided, and the example.bmp file in the zip file. I should be able to create the executable by typing 'make'. The Makefile should also contain a 'clean' target for cleaning up the directory (removing all object files). Make sure you don't include intermediate files: \*.o, executables, \*~, etc., in your submission. (There'll be a penalty for including unnecessary intermediate files).

Please make sure you submit homework before the deadline. There will be a late penalty as per the syllabus.

**If the program does not compile and do something useful when it runs it will not earn any credit.**

[bmplib](#)

[Download bmp lib](#)

(Zip file) - files needed to complete this assignment.