COMP 206 Fall 2018 – Assignment 3

October 22nd, 2018

Objectives:

Build up "systems programming" aspect of C. Work with binary files, read and understand a real-life file-format specification and write C code to interact with binary data. Experience with visual image data. Cute robots. This:

On your marks:

Ensure you're comfortable with the Lecture 10 – 13 slides and videos. Attend Tutorial 3 "C bootcamp". Read ahead on Lecture 14 to understand C programs that use multiple files and make. Read this whole document. Keep the BMP file spec handy: here.

Get Set:

Download A3.zip and unzip. Change to the Assignment3 directory. Type "make test". Ensure you see:

```
$ make test
gcc -g bmp_info.c A3_provided_functions.c A3_solution.c -o bmp_info
gcc  -g bmp_scale.c A3_provided_functions.c A3_solution.c -o bmp_scale
gcc  -g bmp_collage.c A3_provided_functions.c A3_solution.c -o bmp_collage
./bmp_info    rabbit.bmp
UNIMPLEMENTED FUNCTION: bmp_open has not yet been coded. Please complete before submitting!
Error: bmp_info function returned NULL. Cannot report file information.
./bmp_scale   dog.bmp                dog_scaled.bmp 0.5
UNIMPLEMENTED FUNCTION: bmp_open has not yet been coded. Please complete before submitting!
Error: bmp_open returned NULL. Returning from bmp_to_3D_array without attempting changes.
Error: bmp_to_3D_array failed for file dog.bmp.
./bmp_collage robots.bmp             rabbit.bmp     robots_with_one_ear.bmp   80  770 1.0
UNIMPLEMENTED FUNCTION: bmp_collage has not yet been coded. Please complete before submitting!
./bmp_collage robots_with_one_ear.bmp dog.bmp        robots_with_both_ears.bmp 190 205 0.5
UNIMPLEMENTED FUNCTION: bmp_collage has not yet been coded. Please complete before submitting!
```

I recommend that you read carefully through all the files provided. There are 5 ".c" files and 2 ".h" files.

Go!

Type your answers all in the one file: "A3_solution.c". Don't change any of the other files, and only enter code in A3_solution.c within REPLACE EVERYTHING FROM HERE … TO HERE!
You can keep testing your code by typing "make test" after you make changes and save. Or, you can look inside the Makefile and just run each single gcc command yourself if you find this more useful.

The finish line: November 7th, 11:59pm hard deadline

Test at least once on mimi. To visualize your outputs on mimi, take a look at the A3_tester.html. If you place your A3 code within ~/public_html, you can use your web browser to see the files by pointing it to an address like: https://www.cs.mcgill.ca/~dmeger/A3_tester.html (replace with your username).

Hand-in only your A3_solution.c file to My Courses. No extensions will be given for the rest of the term for any problems handing in. Be prepared for both mimi and My Courses to be slow on the day of the deadline. I will only grant exemptions for medical (includes psych), family and religious reasons.

Question #1 – BMP Info (30 marks)

The C file "bmp_info.c" contains a main function that calls the "bmp_open" and "bmp_close" functions and prints the file's information to the terminal if successful. You can compile the program with:

$ gcc bmp_info.c A3_provided_functions.c A3_solution.c -o bmp_info

Test the program with “make test” or:

$ ./bmp_info dog.bmp

Initially, the provided code will report that the function is unimplemented and give an Error. This is normal. You must complete the functions "bmp_open" and “bmp_close” within the file "A3_solution.c" file so that the bmp_info program always outputs the correct data. See the comment in “A3_solution.h” for low-level instructions on what to do with each variable involved.

Notes:

- Many of the variable use “pass by reference with pointers” to effectively return multiple values.
- You must several integer values from the header. I recommend following the example in the slides and in the ExampleCode folder.
- Note especially that bits per pixel is only a 2 byte integer in the BMP file.
- You must use malloc to create space for the image data.
- You must compute the padding yourself.

Testing:

Run bmp_info manually or with “make test” and ensure it matches these precisely:

```
./bmp_info    dog.bmp
bmp_info for file dog.bmp:
  width         = 620
  height        = 712
  bpp           = 32
  padding       = 0
  data_offset   = 138
  The middle pixel has ( row col )=( 310 356 ) and
  color=( 0 0 0 0 ).
```

```
./bmp_info    metro_top_left.bmp
bmp_info for file metro_top_left.bmp:
  width         = 307
  height        = 331
  bpp           = 24
  padding       = 3
  data_offset   = 122
  The middle pixel has ( row col )=( 153 165 ) and
  color=( 10 10 10 ).
```

Question #2 – BMP Scale (30 marks)

The C file "bmp_scale.c" contains a main function that calls the "bmp_scale" function, which is supposed to resize the pixel data of an image by a scaling factor given as an argument. Compile with make or with:

$ gcc bmp_scale.c A3_provided_functions.c A3_solution.c -o bmp_scale

Run the program with:

$ ./bmp_scale dog.bmp dog_scaled.bmp 0.5

Initially, report that the function is unimplemented. You must complete the "bmp_scale" function within the file "A3_solution.c" so that the output image is produced and scaled correctly. As always, details for exactly what is meant by each argument are given in the "A3_solution.h" header file.

Notes:

1. bmp_scale.c uses our helper code to reformat the raw bmp data into a 3D array before calling your code. This is supposed to make life easier. Read the A3_provided_functions .c and .h to understand this and see examples of working with that data.
2. You need to have finished Q1 prior to working on Q2 because bmp_open and bmp_close are called by our helper code.
3. I recommend that you allocate new heap memory for the scaled image. Do not try to do the scaling “in place” or worry about freeing the memory until you get the simplest version working.
4. For the output image to look proper, modify the image header that is passed into your function. Make sure to change overall file size, image width and image height based on the scaled size.
5. You can do all scaling math with simple integer operations (round off downwards). Don’t worry about getting precise as you’re not allowed to use math.h here.

Testing:

1. Open the scaled image. Ensure it looks like the original but has dimensions that have changed by exactly the correct amount. E.g., dog_scaled.bmp should have width 310 and height 356.

Question #3 – BMP Collage (40 marks)

The C file "bmp_collage.c" contains a main function that calls the "bmp_collage" function, which is supposed to open two BMP files, and combine their contents by laying one over top of the other, respecting that the “alpha” channel of the foreground image indicates transparency. Alpha values of 0 mean the background image should show through. Build with “make” or:

$ gcc bmp_collage.c A3_provided_functions.c A3_solution.c -o bmp_collage

Test with “make test”:

$ ./bmp_collage robots.bmp rabbit.bmp robots_with_one_ear.bmp 80 770 1.0
$ ./bmp_collage robots_with_one_ear.bmp dog.bmp   robots_with_both_ears.bmp 190 205 0.5

Initially, the program will print unimplemented. You must complete the "bmp_collage" function within the file "A3_solution.c" so that the output image is produced with the correct contents. I recommend trying to re-use functionality from Q1 and Q2, but here the structure of the function is up to you, so do what you find easiest. Just ensure you match these outputs: