Hack 7.0

Computer Science I

Department of Computer Science & Engineering
University of Nebraska–Lincoln

Introduction

Hack session activities are small weekly programming assignments intended to get you started on full programming assignments. Collaboration is allowed and, in fact, *highly encouraged*. You may start on the activity before your hack session, but during the hack session you must either be actively working on this activity or *helping others* work on the activity. You are graded using the same rubric as assignments so documentation, style, design and correctness are all important.

Exercises

To get more practice working with arrays, you will write several functions that involve operations on arrays. In particular, implement the following functions.

1. Write a function that, given an integer array and an integer x determines if the array contains x anywhere within the array. It should return true if it does, false otherwise.

   ```
   int contains(const int *arr, int size, int x);
   ```

2. Write a function that given an integer array and an integer x determines if the array contains x within two provided indices, i, j. It should return true if it does, false otherwise.

   ```
   int containsWithin(const int *arr, int size, int x, int i, int j);
   ```

3. Write a function that, given an array of integers, its size and a "new size" creates a new deep copy of the array. However, instead of its original size, the new array

should be of the new size. If the new size is less than the old size, only the first `newSize` elements should be copied over. If the new size is greater than the original size, then the new array should be padded out with zeros.

```
int * paddedCopy(const int *arr, int oldSize, int newSize);
```

4. Write a function that, given an array of integers and its size, reverses the elements in the array. For example, if the original array was `[10, 15, 5, 25, 0]` the new array should be `[0, 25, 5, 15, 10]`.

   ```
   void reverse(int *arr, int size);
   ```

5. Write a similar function that creates and returns a new copy of the given array but with its elements in reverse order.

   ```
   int * reverseCopy(const int *arr, int size);
   ```

Image Manipulation

You'll get more practice with 2-dimensional arrays by writing several functions to manipulate images. We've adapted a C image library, the "stb" library, and written several *wrapper functions* to load and save images. Wrapper functions are functions that call other functions but may have some "glue code" to make the control flow or data compatible. In this case, our wrapper functions convert/translate the stb library's image representation into an RGB pixel representation. We've defined a `Pixel` structure that holds 3 integer values one for each of the red, green, and blue color values. We'll cover structures in detail later on, but it won't prevent you from working with them.

You can declare and use a `Pixel` type like you would an `int` or `double`:

```
1  //a single pixel:
2  Pixel p;
3  //an array of n pixels:
4  Pixel *p = (Pixel *) malloc(sizeof(Pixel) * n);
5
6  //swap two pixels:
7  Pixel a, b;
8  ...
9  Pixel temp = a;
10 a = b;
11 b = temp;
```

An $h \times w$ (height by width) image can be represented as a two dimensional array of `Pixel` types; in particular: `Pixel **image`. Everything we've covered using two dimensional arrays of `int` types applies to `Pixel` types.

We've provided a library of functions to load and save a file (you'll need to RTM) and specified several function signatures for functions you need to implement.

- `copyImage()` should produce a deep copy of the given image.
- `flipHorizontal()` should flip the image horizontally as depicted in Figure 2.

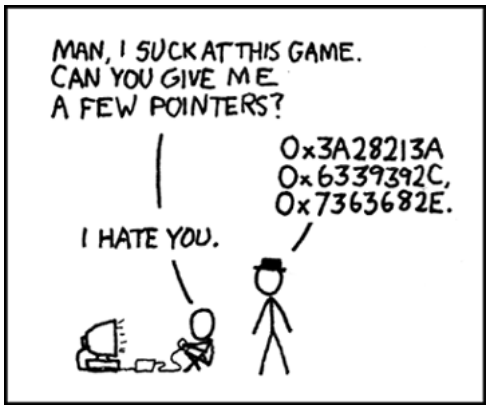

Figure 1: Original Image

Figure 2: Flipped Horizontally

- `flipVertical()` should flip the image vertically as depicted in Figure 3.
- `rotateClockwise()` should produce a new image that is rotated 90 degrees clockwise. This function must produce a new image because an $h \times w$ sized image that has been rotated will be a $w \times h$ image. This operation is depicted in Figure 4.

Instructions

- For the warm-up, place all your function prototypes into a file named `array_utils.h` and and their definitions in a file named `array_utils.c`. You will need to turn these in via webhandin.

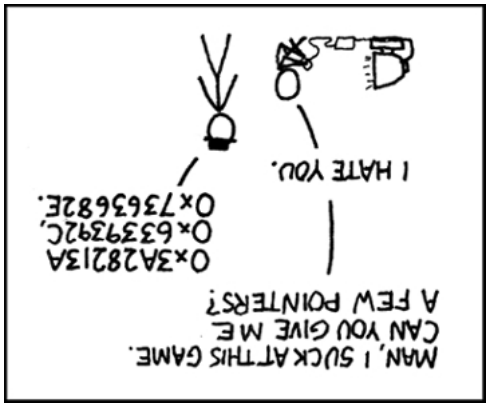

Figure 3: Flipped Vertically

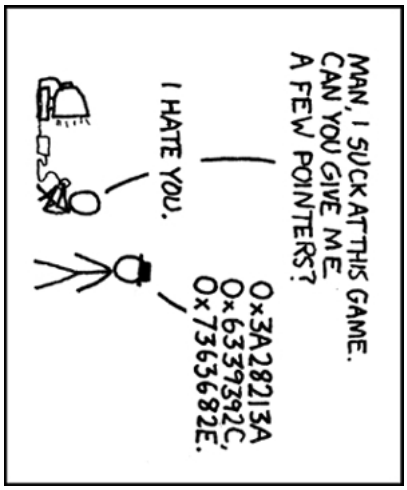

Figure 4: Rotated Clockwise

- In addition, you'll want to create a main test driver program that demonstrates at least 3 cases per function to verify their output. You need not hand in this test file, however.
- For the Image Manipulation section, you can use the starter code provided here:

  https://github.com/cbourke/CSCE155-Hack7.0

  However, you only need to handin `imageUtils.h` and `imageUtils.c`
- You should test all your functions with an image (load it, manipulate it and save it) of your choice.
- As a first step, you should add documentation to all your functions. Use this as an opportunity to discuss how the functions should work and to *whiteboard* your designs and solutions with other students.
- You are encouraged to collaborate any number of students before, during, and after

your scheduled hack session.

- You may (in fact are encouraged) to define any additional "helper" functions that may help you.
- Include the name(s) of everyone who worked together on this activity in your source file's header.
- Turn in all of your files via webhandin, making sure that it runs and executes correctly in the webgrader. Each individual student will need to hand in their own copy and will receive their own individual grade.