John Sullivan
16 February 2020
Prof. Herve
CSC 412

# ***Report for Assignment 02***

## Design Choice

For version one of the programs, I chose to make a data structure that held the location image data, and then stored each one of those structures in a list. The structure held the x and y values as integers. I chose integer because it seemed like the most practical way to store a coordinate. I didn't think I'd be getting any arbitrarily large or small (huge negative number) as coordinates so integer seemed fine. The data structure held the name of the location as a char* because that's the easiest way to store a string, and the index was also stored as an integer. I didn't use unsigned integer even though I knew I'd never have a negative index, but I also knew I'd never get an extremely large index either. I wouldn't expect over 100,000 inputs to this program, but maybe that is a limitation.

For the second version of the programs, I decided to make another type called LocNameList which would be initialized as a list and store in each different data structure: a location name and a list of images associated to it as well as a counter for how many images are stored in each location. The reason I chose to keep a counter for the images was to be able to print them when requested in a query, with a stopping point for the loop I'd print in. I also kept an int* which stored the length of my locations list, which wasn't part of the structure but was incremented every time I added a new location to the list. This was useful to pass around and hold a value in memory, as I needed it in several functions and couldn't return more than one thing at a time. Instead of holding copies of images in my new list of structs here, I decided to hold pointers to the images instead. This was beneficial for memory allocation, as it can be dangerous to store copies of structs in two different places.

## Limitations

If an input size of more than 1000 images are entered, the formatting of the output starts to look funky, because of the specifications of the assignment say to only format the image # to be only 3 digits. My program will not run correctly if you give it an input size too large. When I say too large, I mean more than 3000+ values. The string will be too long, and I don't have any handling in my bash scripts to allow for larger strings to be input via the command line. This limitation lies in not my program but the max string length of bash itself, so I can't do anything to change that without actually modifying the host machine's maximum memory allocation in bash for string input. But this is fine, because the input size will never be larger than 1000 for grading purposes. I don't believe there are any other limitations to my program, as I test for valid input in several places as well as in the bash scripts.