CMSC 216 Exercise #4 Spring 2019

PhotoAlbum **Due: Wednesday March 27, 2019, 11:30PM**

1 Objectives

To practice dynamic memory allocation.

2 Overview

The first thing you need to do is to copy the directory photoalbum we have left in the grace cluster under the exercises directory. Remember that you need that folder as it contains the .submit file that allows you to submit.

3 Specifications

For this exercise you will implement functions that support a photo album application. The prototypes for the functions can be found in the photoalbum.h file.

1. Photo *create_photo(int id, const char *description) - Returns a dynamically-allocated Photo structure initialized based on the provided parameters. If the description parameter is different from NULL, the function will dynamically allocate memory for the description and copy the description. If description is NULL, no memory allocation will take place and the description field will be initialized to NULL. The function will return NULL if a memory allocation fails. You don't have to worry about freeing memory if any memory allocation fails (e.g., one memory allocation was successful, but a second one fails).

2. void print_photo(Photo *photo) - Prints a photo id and the description. If the description is NULL, the message description message will be "None". The function will perform no task if the photo parameter is NULL. See the public tests for information regarding output format.

3. void destroy_photo(Photo *photo) - Deallocates any dynamically-allocated memory associated with the photo parameter. The function will perform no task if the photo parameter is NULL.

4. void initialize_album(Album *album) - Initializes the album size to 0. You can assume this function will not be called on an album that has already been initialized. The function will perform no task if the album parameter is NULL.

5. void print_album(const Album *album) - Prints the contents of the album. If the album has no photos the message "Album has no photos." will be printed. The function will perform no task if the album parameter is NULL. See the public tests for information regarding output format.

6. void destroy_album(Album *album) - Deallocates any dynamically-allocated memory associated with the album and sets the album size to 0. The function will perform no task if the album parameter is NULL.

7. void add_photo_to_album(Album *album, int id, const char *description) - Appends (to the end of the array) a photo if there is enough space (if the album size is less than MAX_ALBUM_SIZE). No photo will be added if a photo cannot be created. The function will perform no task if the album parameter is NULL.

You may want to take a look at the public tests in order to understand the functionality associated with the functions above.

4 Requirements

1. It is your responsibility to verify that your program generates the expected results in the submit server.
2. Your code must be written in the file photoalbum.c.
3. Do not add a main function to the photoalbum.c file.
4. Use the provided makefile to build public tests.
5. All your C programs in this course should be written using the compiler `gcc`, with the options defined in the gcc_aliases_info.txt file. This file can be found in the info folder of the public grace account.
6. Your program should be written using good programming style as defined at

   http://www.cs.umd.edu/~nelson/classes/resources/cstyleguide/
7. You just need to implement the functions described above. You may add additional functions if you want, but define them as static.
8. You may not change the photoalbum.h file provided.
9. You are encourage to define your own tests (files similar to the public01.c, public02.c, etc. files provided).
10. You don't need to use the macros SUCCESS and FAILURE you will find in the photoalbum.h file.
11. Using valgrind to detect memory problems can be very helpful.
12. To check for memory problems you either use our memory tool (my_memory_checker) or valgrind, but not both. If you want to run valgrind you must comment out the function calls start_memory_check() and stop_memory_check() you will find in the public tests.

5 Submitting your assignment

1. In the assignment directory (photoalbum) execute the command **submit**.
2. Your assignment must be electronically submitted by the date and time above to avoid losing credit. See the course syllabus for details.

6 Grading Criteria

Your assignment grade will be determined with the following weights:

Results of public tests	70%
Results of release tests	30%

Notice that even though we will not look at your code, we expect good style in your code.

7 Academic integrity statement

Please **carefully read** the academic honesty section of the course syllabus. **Any evidence** of impermissible cooperation on assignments, use of disallowed materials or resources, or unauthorized use of computer accounts, **will be submitted** to the Student Honor Council, which could result in an XF for the course, or suspension or expulsion from the University. Be sure you understand what you are and what you are not permitted to do in regards to academic integrity when it comes to assignments. These policies apply to all students, and the Student Honor Council does not consider lack of knowledge of the policies to be a defense for violating them. Full information is found in the course syllabus– please review it at this time.