CMSC 216 Exercise #3 Summer 2022

Photo Album **Due: Wednesday June 29, 2022, 11:55 pm**

1 Objectives

To practice dynamic memory allocation.

2 Overview

The first thing you need to do is to copy the directory photo_album 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.

You can discuss this exercise with other classmates, but you may not exchange any code or write code together.

3 Grading Criteria

Your assignment grade will be determined based on the following weights:

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

4 Academic integrity statement

Please **carefully read** the academic honesty section of the course syllabus. We take academic integrity matters seriously. Please do not post assignment solutions online (e.g., Chegg, github) where others can see your work. Posting code online can lead to an academic case where you will be reported to the Office of Student Conduct.

5 Specifications

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

6 Requirements

1. Use the C code development strategy described at `http://www.cs.umd.edu/~nelson/classes/resources/cdebugging/development_strategy/`.
2. Your code must be written in the file photo_album.c.
3. Do not add a main function to the photo_album.c file.
4. Use the provided makefile to build public tests.
5. Your program should be written using good programming style as defined at

   `http://www.cs.umd.edu/~nelson/classes/resources/cstyleguide/`
6. You just need to implement the functions described above. You may add additional functions if you want, but define them as static.
7. You may not change the photo_album.h file provided.
8. You are encourage to define your own tests (files similar to the public01.c, public02.c, etc. files provided).
9. You don't need to use the macros SUCCESS and FAILURE you will find in the photo_album.h file.
10. To check for memory problems you can use the CMSC216 memory tool (my_memory_checker), valgrind, or gcc -fsanitize=address. Remember to only use one at a time as you may get confusing information. If you want to disable the CMSC216 memory tool in the public tests, comment out the function calls start_memory_check() and stop_memory_check().
11. A common error is for students to first allocate memory and then check for parameters' validity. You should always first check for parameters' validity before allocating memory.

7 Submitting your assignment

Submit as usual.