Photo Album **See Piazza for due date**

1 Objectives

To practice dynamic memory allocation.

2 Overview

First, copy the directory photo_album from the grace 216public exercises directory. As usual, that folder contains the .submit file that allows you to submit.

You can discuss this exercise with your classmates, but you may not exchange any code nor 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 are in file photo_album.h.

1. Photo *create_photo(int id, const char *description)

   Returns a dynamically-allocated Photo struct initialized based on the provided parameters. If parameter description is not NULL, assume it points to a nul-terminated string. In this case, the function dynamically allocates memory to hold description's string and sets the Photo struct's description field to point to this allocated memory. If parameter description is NULL, no memory allocation takes place and the Photo struct's description field is initialized to NULL. The function returns 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 is successful, but a second one fails).

2. void print_photo(Photo *photo)

   Prints the photo's id and description. If the description is NULL, the message description message is ''None''. The function does nothing if the photo parameter is NULL. See the public tests for information regarding output format.

3. void destroy_photo(Photo *photo)

   Deallocates all dynamically-allocated memory associated with parameter photo. The function does nothing if parameter photo is NULL.

4. void initialize_album(Album *album)

Initializes the album size to 0. Assume this function is not called on an album that has already been initialized. The function does nothing if parameter album 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." is printed. The function does nothing if parameter album is NULL. See the public tests for information regarding output format.

6. void destroy_album(Album *album)

   Deallocates all dynamically-allocated memory associated with parameter album and sets the album's size to 0. (The Album struct itself is not deallocated, so "clear_album" may be a better name for this function.) The function does nothing if parameter album 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 space (if the album size is less than MAX_ALBUM_SIZE). photo is added if a photo cannot be created. The function does nothing if parameter album is NULL or the album has no space for the photo.

You want to 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 only have to implement the functions described above. You can have additional functions, but define them as static.
7. Do not change the photo_album.h file provided.
8. You are encouraged 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 defined in file photo_album.h.
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, otherwise 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().

7 Submitting your assignment

Submit as usual.