COMP 2160 Programming Practices
Assignment 4

Due Date: Friday, December 7 at 11:59 pm

Objectives

- Garbage Collection
- This is a detailed assignment so rather than providing the details here, most of the details will be discussed in class.

Summary

In this assignment, you are to implement a simple object manager. An object manager is another form of table, but we now store generic objects in a managed buffer (the internal implementation of a table varies based on need). Using a buffer means that we must directly manage the memory and references to objects. This means that our object manager must implement reference counting and a garbage collector, so that we properly handle creation and deletion of objects. The object manager's interface is given by the file `ObjectManager.h` (posted in the Assignments section) and the implementation will be in the file `ObjectManager.c`. Your task is to implement the functions required for the object manager. This includes all the functions listed in `ObjectManager.h`, along with any (private) static functions that will be useful. You will also need to define appropriate data types and data structures for keeping track of the objects that have been allocated. You are NOT allowed to change the prototypes of the functions that are provided to you in `ObjectManager.h`. To summarize, the functionalities you have to implement are:

- `initPool()` - Initialize the object manager upon starting.
- `destroyPool()` - Clean up the object manager upon quitting.
- `insertObject(size)` - request a block of memory of given size from the object manager.
- `retrieveObject(id)` - retrieve the address of an object, identified by the reference id.
- `addReference(id)` - increment reference count for the object with reference id.
- `dropReference(id)` - decrement reference count for the object with reference id.
- `compact()` - initiate garbage collection (see below). Note this function is NOT part of the interface available to client code.
- `dumpPool()` - print (to `stdout`) info about each object that is currently allocated including its id, start address, size.

A more thorough discussion of the functionality of these routines will be discussed in class.

Garbage Collection

You will implement a Mark and Sweep Defragmenting/Mark-compact garbage collector, as described in class. This function is implemented by `compact()` in the object manager. So that we can evaluate your implementation, every time the garbage collector runs print out the following statistics (to `stdout`):

- The number of objects that exist
- The current number of bytes in use
- The number of bytes collected

Data Structures

You must manage the tracking of allocated objects using an index, which will be implemented using a linked list. Each node in your linked list contains the information needed to manage its associated object. Note that the index is implemented inside the object manager, you don't need any additional files.

Testing your Object Manager

You will (eventually) be provided with several files (containing the `main()` function) for testing your object manager (you can start with `main0.c` on the web site). In the meantime, you'll have to use some of your new found testing and debugging skills to ensure your code can handle anything a program may throw at it -- at a minimum, the code you hand in must use assertions, etc as required for assignments 2 and 3. Note that the main files will arrive very late in the game. If you wait until they're available there's no way you'll finish the assignment on time.

Other Comments

- Try to start the assignment as soon as possible.
- Test each feature as it is implemented. Create your own test cases to test your code.
- You will have multiple files so a Makefile is required. Include a rule for `main.c` so that you and the marker can swap in different main files by renaming and touching the files.
- Although you are not required to do so, you should try profiling your code with gprof to determine the code's hot spots.

Hand-in Instructions

Create a directory called `comp2160-a4-<yourlastname>-<yourstudentid>`, and place your `.c` files inside the directory.

Then run the command:

```
handin 2160 a4 comp2160-a4-<yourlastname>-<yourstudentid>
```

- You may *optionally* include a README file in your directory that explains anything unusual about compiling or running your programs.
- You may resubmit your assignment as many times as you want, but **only the latest** submission will be kept.
- We only accept homework submissions via handin. Please **do not** e-mail your homework to the instructor or TAs – it will be ignored.
- You **must** submit an Honesty Declaration (digitally in UMLearn). Assignments submitted without a corresponding Honesty Declaration **will not be graded**.
- We reserve the right to refuse to grade the homework, or to deduct marks if these instructions are not followed.
- **WE WILL STRICTLY ENFORCE THE DEADLINE FOR HW4. THE HANDIN FOLDER WILL BE CLOSED AFTER THE DEADLINE. IF YOU DO NOT SUBMIT BY THEN, YOU WILL RECEVIE 0 FOR HW4.**