CS 214 Homework 4

Fall 2021

Introduction

In this assignment, you'll implement a custom version of malloc, free, and realloc. You should implement these in `mymalloc.c` and `mymalloc.h`. These files should not contain a `main` function, so you may want a separate `.c` file for testing.

You can work in small groups (1 – 3 people). Please include a README file with your code that contains all partners' names and netIDs. Only one person from each group should submit the assignment.

Library functionality

You should implement these functions, described below:

```
void   myinit(int allocAlg);
void*  mymalloc(size_t size);
void   myfree(void* ptr);
void*  myrealloc(void* ptr, size_t size);
void   mycleanup();
```

- `myinit(allocAlg)`

  Create a 1 MB "heap" and perform any other initializations your code needs. You can assume any application using your library will call this first.

  The `allocAlg` argument describes what algorithm to use to find a free block:

  0: first fit
  1: next fit
  2: best fit

- `mymalloc(size)`

  From the "heap", allocate a region of at least the requested `size` and return a pointer to the beginning of the region. If it cannot be allocated, return `NULL`.

  All returned addresses must be 8-byte aligned. That is, the region you allocate should start at an address that's divisible by 8.

  If `size` is 0, `mymalloc` does nothing and returns `NULL`.

- `myfree(ptr)`

  Mark the given region as free and available to be allocated for future requests. It should be coalesced with adjacent free regions.

  You should maintain an explicit free list.

  If `ptr` is `NULL`, `myfree` does nothing.

- `myrealloc(ptr, size)`

  Reallocate the region pointed to by `ptr` to be at least the new given `size`. If this cannot be done in-place, a new region should be allocated, the data from the original region should be copied over, and the old region should be freed.

  If the reallocation can't be done, return `NULL`.

  If `ptr` is `NULL`, this is equivalent to `mymalloc(size)`.

  If `size` is 0, this is equivalent to `myfree(ptr)` and `myrealloc` returns `NULL`.

  If both `ptr` is `NULL` and `size` is 0, `myrealloc` does nothing and returns `NULL`.

- `mycleanup()`

  Free the 1 MB "heap" and perform any other cleanup your code needs. You can assume any application using your library will call this last.

  Your library should support "resetting" everything by calling `mycleanup` followed by `myinit`.

Note: outside of `myinit` and `mycleanup`, you should not call the standard `malloc`, `calloc`, `free`, or `realloc` functions.

Compiling and testing

You should create a Makefile so that running `make` or `make all` builds your program. You should use similar `CFLAGS` to `gcc` as in previous homeworks, e.g.:

```
-g -Wall -Wvla -fsanitize=address
```

Submission

If you develop on your local machine, please be sure to test your code on ilab before submitting.

Please submit the assignment on Canvas as a tar file `hw4.tar` that, when expanded, produces a `hw4` directory (possibly with additional .c/.h files):

```
hw4
├── Makefile
├── mymalloc.c
├── mymalloc.h
└── README.txt
```