# Computer Systems
# Project 4
# Fancier allocators

## 1 Preparing for mark-sweep garbage collection

For this project and the next one, we will be working with a *best fit* allocator. It keeps a doubly linked-list of free blocks; when an allocation is requested, it searches that list for the closest fit. In the absence of an acceptable free block, it pointer-bumps to expand the heap.

Part of the goal of this project is to understand a slightly more realistic allocator that reuses deallocated blocks. A secondary goal is future-oriented: we will need an allocator suited to a *mark-sweep garbage collector*—a kind of garbage collector that we will implement in Project 5. Thus, we will be modifying this *best fit* allocator to prepare it for use in the next project.

## 2 Getting started

### 2.1 Creating the repository

1. **Login to the server:** Connect to the course server.
2. **Login to GitLab:** From your browser, login to `https://gitlab.amherst.edu`
3. **Start a new project:** On the top toolbar of the GitLab window, click the little drop-down menu marked by a plus-sign. Select `New project`. Set the *Project name* to be `sysproj-4`, and leave the other default values. Click on the `Create project` button at the bottom.
4. **Clone the repository onto the course server:**

```
$ git clone git@gitlab.amherst.edu:yourusername/sysproj-4.git
$ cd sysproj-4
```

5. **Download the source code:** After you download the files, use `ls -l` to list the directory and see what you have.

```
$ wget -nv -i https://bit.ly/cosc-171-24s-p4
$ ls -l
```

6. **Add/commit/push the source code to the repository:**

```
$ git add *
$ git commit -m "Starting code."
$ git push
```

## 2.2 Compiling and running

This collection of code works identically to the code from Project-3. To compile the pieces:

```
$ make clean
$ make libbf memtest
```

Then, to run `memtest` (or anything else) with your allocator code, and then turn off your code:

```
$ export LD_PRELOAD=${PWD}/libbf.so
$ ./memtest
$ unset LD_PRELOAD
```

# 3 Your assignment

## 3.1 Part I: Commenting the code

Open `bf-alloc.c`. Its basic structure matches that of `pb-alloc.c` from the previous project. There are some key differences, including the definition of the `header_s`, which now contains additional fields for organizing each block.

Notice that `malloc()` and `free()` are devoid of comments. **Comment these functions**, providing a guide to any reader of the code as to what is happening in each group of lines. As before, you may collaborate freely with others in figuring out the code and writing these comments.

## 3.2 Part II: Make it align

This allocator does **not** provide double-word aligned blocks in the way that it should. **Port your code from Project-3**, making the blocks that are created by pointer-bumping be properly aligned. Note that the contents of the headers may be different in this allocator, so be sure that your alignment code will work with that change.

## 3.3 Part III: Create an *allocated list*

For the initial, downloaded code, blocks are only ever linked into a *free list*; that is, when a block is allocated, it isn't on a list at all; the caller to `malloc()` is responsible for keeping track of the block until it calls `free()` to return the block to the allocator. Under *explicit memory management*, this is the normal state of affairs.

A garbage collector, however, must be able to find all the allocated blocks in order to determine which are *live* (still usable by the program) and which are *dead* (no longer usable). In preparation of using this allocator as part of a garbage collector, modify this *best fit* allocator to **keep a linked list of allocated blocks**. When a block is allocated, add it to this list; when freed, remove it from this list before adding it to the free list.

### 3.4 Part IV: Test your code

**Modify the testing program**, `memtest`, to do some allocations, deallocations, and reallocations. Make sure that it still works properly with all of your code changes.

## 4 How to submit your work

First, be sure that the most recent versions of your work are up-to-date on the GitLab server by performing an *add/commit/push* with `git`. Then, go to GitLab with your browser, and add me (*sfkaplan*) as a *Developer* to your repository.

**This assignment is due on Sunday, Mar-03, 11:59 pm.**