Lab 7: Implicit Free List Memory Allocator (29.2.16)

In this lab, you will get familiar with the implicit free list memory allocator and make relatively simple change to the design. To get checked off on this lab, you need to complete Exercises 1 and 2 below. After that, you should start working on the programming project where you will be writing your own dynamic memory allocator.

Getting started

1. Copy all the files in lab folder to a protected directory in which you plan to do the work.
2. Type you name and email address in the header comment at the top of `mm.c`.
3. Type the command make to compile and link a basic memory allocator, the support routines, and the test driver. This basic memory allocator is based on an implicit free list, first fit placement, and boundary tag coalescing.
4. Run the test driver `mdriver` to test the memory utilization and throughput performance of this basic memory allocator. It'll take a minute to run. You should see the following performance index printed:

```
Perf index = 29/40 (util) + 1/60 (thru) = 30/100
```

See `proj2-handout` for details on how these scores are computed. The memory utilization aspect (`util`) carries a 40% weight. The throughput aspect (`thru`) carries a 60% weight. As you can see, this basic memory allocator does not earn a very high performance index.

Exercise 1

The allocator we have provided performs a first-fit search of the implicit free list. Develop a version of the allocator that performs a next-fit search instead of a first-fit search. Run the test driver mdriver. Use the –v option to print the per-trace performance breakdowns for the version of the allocator that performs a next-fit search.

Exercise 2

Develop a version of the allocator that performs a best-fit search instead of a first-fit search. Develop a version of the allocator that performs a next-fit search instead of a best-fit search. Run the test driver mdriver. Use the –v option to print the per-trace performance breakdowns for the version of the allocator that performs a best-fit search.

Project 2 - DIY allocator

The implicit free list memory allocator we have provided in mm.c is a functionally correct but very poorly performing malloc package. In the project, you will explore the design space creatively and implement the best memory allocator that you can. It may be done individually or in pairs. Read proj2-handout.pdf for the project description, details on how your grade will be calculated, programming rules, and other helpful hints.

The project will be submitted via moodle. Due date is 21:55 hours, Thursday, March 31.
