Lab 7 – Implicit Free List Memory Allocator (17.2.14)

In this lab, you will get familiar with the implicit free list memory allocator and make a relatively simple change to the design. After that, 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 team member names and email addresses 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 `project1-malloc` 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.

Boundary tag optimization

Carefully go through the source code provided in `mm.c`. The `mm.c` file implements a simple memory allocator as described in Section 10.9.12 of textbook. It requires both a header and a footer for each block in order to perform constant-time coalescing. Modify the allocator so that free blocks require a header and footer, but allocated blocks require only a header. Use the driver program to test the modified allocator. Your implementation must pass the correctness tests performed by the driver program.

Project malloc

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. The project will be graded out of 100 points. It may be done individually or in pairs. Read `project1-malloc.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, Monday, March 3th.