CSE320 : System Fundamentals II Assignment 4

DUE: 30-Oct-2024

Dynamic Memory Allocation

Overview/Goals

This assignment will give you experience with:

1. Understanding how dynamic memory management schemes work and how free and allocated block lists are maintained.
2. Understanding block splitting and coalescing during allocation and free operations
3. Understanding how heap extension works

Basic Program Specifications

Incomplete memory management code has been provided with this assignment. Complete the code by implementing the TODO sections in the comments.

1. **Initialize Heap –** The code will be able to initialize the heap with prologue and epilog blocks.
2. **Allocation** – The code should be able to allocate blocks, split blocks that are too large, and, if needed, extend the heap space.
3. **Free –** The code should be able to free allocated blocks and coalesce adjacent free blocks.

Tasks

1. Download malloc.zip provided in the assignment
2. Complete the TODO sections
3. Build, test, and debug the resulting code

Building the Code

The code includes a main function in **app.c** which calls a unit test in **unit_test.c.** Build the code with the provided **Makefile** (type 'make' on the unix command line). This will build app.

If you need to debug code with gdb, then, on the Unix command line, type 'make debug'. You can then run app under gdb and use the debugger.

Sample Output

When completed the code should produce output that looks approximately like the following:

```
malloc ptr[0] = 0x7f362191f030
malloc ptr[1] = 0x7f362191f430
malloc ptr[2] = 0x7f362191f830
malloc ptr[3] = 0x7f362191fc30
malloc ptr[4] = 0x7f3621920030
malloc ptr[5] = 0x7f3621920430
malloc ptr[6] = 0x7f3621920830
malloc ptr[7] = 0x7f3621920c30
malloc ptr[8] = 0x7f3621921030
malloc ptr[9] = 0x7f3621921430
free ptr[1] = 0x7f362191f430
free ptr[3] = 0x7f362191fc30
free ptr[5] = 0x7f3621920430
free ptr[7] = 0x7f3621920c30
free ptr[9] = 0x7f3621921430
p:0x7f362191f430 == ptr[1]:0x7f362191f430
p:0x7f3621921430 == ptr[9]:0x7f3621921430
p:0x7f362191fc30 == ptr[3]:0x7f362191fc30
p:0x7f362191fe40 > ptr[3]:0x7f362191fc30
p:0x7f362191fe40 < ptr[4]:0x7f3621920030
SUCCESS!
```

Specific addresses may vary but there should be no diagnostics from the malloc unit test code like:

```
Incorrect size
Incorrect inuse
Inconsistent header and footer
Unexpected malloc
```

The message at the end of the run should indicate
SUCCESS!
As it does in the sample above.

Reflection
Write a very brief reflection on your work for this assignment.
Did you find the embedded instructions and hints in the source code helpful or confusing? What section(s) of the code were most difficult to complete?
Did this assignment clarify some of the memory management concepts discussed in the lecture?

Submission Instructions

Deliverable Files: All source code and Makefile packaged in a single zip or tgz (gzipped tar) file.

Please follow this procedure for submission:

1. Place the deliverable files into a folder by themselves. The folder's name should be CSE320_HW4_<yourname>_<yourid>. So if your name is Alice Kim and your id is 12345678, the folder should be named 'CSE320_HW4_AliceKim_12345678'

2. Compress the folder and submit the zip file.
    a. On Windows, do this by clicking the right-mouse button while hovering over the folder. Select 'Send to -> Compressed (zipped) folder'. The compressed folder will have the same name with a .zip extension. You will upload that file to the Brightspace.
    b. On Mac, move the mouse over the folder then right-click (or for single button mouse, use Control-click) and select **Compress**. There should now be a file with the same name and a .zip extension. You will upload that file to the Brightspace.
3. Navigate to the course Brightspace site. Click **Assignments** in the top navbar menu. Look under the category 'Assignments'. Click **Assignment4.**
    a. Scroll down and under **Submit Assignment**, click the **Add a File** button**.**
    b. Click **My Computer** (first item in list).
    c. Find the zip file and drag it to the 'Upload' area of the presented dialog box.
    d. Click the **Add** button in the dialog.
    e. You may write comments in the comment box at the bottom.
    f. Click **Submit.** 🡐 Be sure to do this so I can retrieve the submission!

Grading Criteria

The assignment will be graded for the following items:

Code builds without error or warning [50% of points]

Code works and provides correct results without crashing (Incorporates proper use of C data types, pointers, arrays, structures, and dynamic memory allocation.)
[30% of points]

Added code is well commented and consistently formatted (readability of the code)
[10% of points]

Assignment submission follows all directives (includes required files, Makefile, etc., is backaged in a zip file…) [5% of points]

Quality and completeness of report [5% of points]