EE 312
Assignment 6

Dynamic Memory Allocation

Executive Summary
In this lab, you will learn about the internals of malloc and free by implementing your own memory allocation and free functions.

Purpose
- Increase your understanding of dynamic memory allocation
- Improve your pointer manipulation skills

Deliverables
In this assignment, you will be implementing a Knuth heap. You will be designing your own malloc and free functions.

Knuth Heap
A block in the knuth heap is made up of three parts. Two metadata and the actual data. It looks like the following.

Metadata
Data
Metadata

The metadata hold an integer that specifies the size of the "data" in words. One word is 4Bytes (remember: one char is one Byte). The metadata is a positive number if the whole block is not occupied. It is negative otherwise.
Each metadata takes up two words in terms of space.
For example:

+2
Data
+2

In this example, the metadata is 2, which means that the size of data is 2 words (8Bytes). Since the metadata is a positive integer, this means that the entire block is free and it can hold up to two words. Metadata cannot be use to store data. They are only used to indicate whether a block is free or used, and to specify the length of the data.

Another example:

-4
Data
-4

In this example, the metadata is -4. This means that the size of data is 4 words and that the block is used.

The Assignment

Init
You are given an initial char array called slab that contains 10,000 bytes. The slab array is initialized as follows. You must initialize the heap before calling malloc or free. After initializing the heap, it should look as follows.

+2496
Data
+2496

Malloc
Your malloc function takes the number of bytes to allocate as a parameter and returns a pointer to the beginning of the data part.

-2
Data
-2

You can only allocate memory in multiple of words. For example, if the user wishes to allocate 10Bytes, your malloc function should reserve 3 words in your array (12 Bytes). After reserving the space in the heap (slab array) you must update the metadata correspondingly.
my_malloc needs to check in the entire slab array if there is an available block of contiguous bytes and return the first block that can satisfy the user request. You want you to find the first fit rather than the best fit.
When allocating memory on the heap, you must start allocating from the end of the heap.
If a block cannot be reserved, my_malloc should return null. It may be possible that the sum of available bytes is greater than the requested number of bytes, but no single contiguous available block is able to satisfy the request. In which case, your my_malloc function should return NULL.

Free

You're my_free function should free a reserved block in the heap and update the metadata accordingly. The function takes a pointer to the beginning of the data part and frees the whole block.
You won't call my_free() without a valid pointer from a my_malloc() call.

Deliverables
Your given a main.c file and an assignment-6.c file. You are responsible for complete the three functions in the assignment-6.c file (init_heap, my_free, my_malloc)
You will need to submit the main.c file and the assignment-6.c file to assembla.

FAQ

1. **How do malloc and free actually work?** http://stackoverflow.com/questions/1119134/how-do-malloc-and-free-work In particular, real-world implementations use advanced data structures to speed up the search for the right sized block. Furthermore, the malloc library code calls sbrk (in UNIX) when it no longer has enough memory to satisfy a call.

2. **How can I store an int in a char array for the metadata?**
   Trick your program into thinking it can store an int. Think casts.

3. **Why is a char* passed in to free?**
   A char is 1 byte so you can use char* to point to any byte in memory