Machine Problem 1 Report

Authors: Clifton Sims, Chris Comeaux

Design Description

We designed our memory allocator using the notion of a Singly Linked List. To do this, we first allocated a block of memory using the function malloc() and kept track of the memory by using pointers. We then created a structure named Node to represent a Link List node. Node held the next pointer, the key, and the value length. After adding a certain node to the allocated block of memory, we then inserted the value that corresponds to node's key directly after the node, creating a single node in memory.

Questions

There indeed is a wastage of memory in our program. Even though we say we "deleted" a node, we actually did not delete it. Instead, we simply break the chain so that it is not a part of the linked list, but still exist in the memory. In other words, the deleted node is still sitting in the allocated memory without any way of being accessed, therefore that block of memory is wasted. Our program can avoid this wastage through a clever use of the data in the deleted nodes. The idea we came up with is that we would have a free-head pointer, kind of like how we had one for the beginning of the Link list, but this pointer would point to the first node that was 'deleted' (still exist in memory) and would start a new linked list of deleted nodes. If more nodes are deleted, then the they are inserted like they would for a regular linked list. Finally, when the memory becomes full, if there are any deleted nodes in this "free" linked list of deleted nodes, the insertion of the new nodes would overwrite the value where the free-head pointer is pointing to. Then the free-head pointer would point to the next earliest delete node. With this implementation, there will never be a time where there is space in memory that we cannot insert a node to. If we did not use the "free" linked list implementation, then things would be different. For example, if the freeptr and the tailptr reach the end of the allocated block and we delete a node somewhere within the block, then there is no way to access that free space and insert a new node into that space.

The max size for the value of a node would be given by: max_value_size = node_size - pointers(8 bytes) - key (4 bytes) - value_length(4 bytes) where node_size is decided beforehand. For example, if node_size = 128 bytes then the max size for the value would be 112 bytes.