Kerry Zeng, Section 503, and Cesar Ortega, Section 501

Machine Problem 1

Objective:

The objective of this assignment was to provide a review on pointers and memory allocation through the implementation of a linked list. By completing this we should have a better understanding of pointer arithmetic and memory allocation.

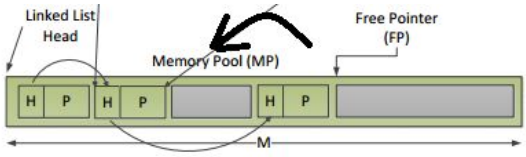
Implementation:

void Init(int M, int C)	Allocates a memory block of size M bytes and initializes some global variables. Of note, the head and tail pointers, as well as numbers to keep track of the size of the nodes (int C) as well as how many nodes may fit in the memory.
void Destroy()	Calls free() on the head pointer, freeing up the memory pointed to by head.
int Insert(int x, char* value_ptr, int value_len)	Checks if the value is too big or If the list is full and returns an error if either of these cases is true. Then, it will attempt to insert the node into the allocated memory by using a pointer and initializing the next, key, and size member variables. It will then use memcpy to copy over the value given by the "char* value_ptr" input. Then the free pointer is moved by the node size amount given (default 128).
char* Lookup(int x)	Iterates over the array to find input int x and returns a char* to the address of the key that matches it.
void Delete(int x)	Iterates over the array until it finds the node to be deleted. Once found, it copies the data of the next nodes after the note with the key. After it's done copying, it moves the free pointer and the tail.
struct Node()	A simple struct with the member variables : Node* next, int key, int size, int val. It's used to store the pointer to the next node as well as the payload.

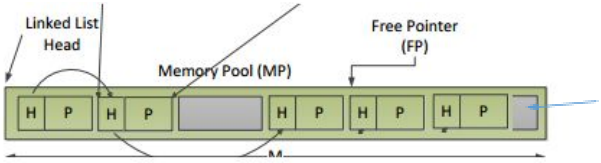
Questions:

- Do you notice any wastage of memory when items are deleted?
  Yes. The initial design wasted memory because the "deleted" node would still exist, but the node previous would point to the next node in memory. This would simulate it being deleted, but it would still take up space. However, we completed the bonus so that it will shift all the nodes over, thus taking advantage of the freed up space after deletion.
- If so, can your program avoid such wastage? How would you do so?

  We can reformat the array so that it doesn't leave the node there. We did this by shifting all the nodes after the deleted ones left by one so that the next node would take the position of the previous one.



- Can you think of a scenario where there is space in the memory but no insertion is possible?
  Yes. For example, say there was 40 bytes left, but we're trying to insert a value that's 50 bytes, it wouldn't work because you would be trying to write outside of the allocated memory.



- What is the maximum size of the value when the pointers are 8 bytes?
  The maximum size of the value when pointers are 8 bytes would be a formula of *node size - 2*size of an int - 8* where size of an int varies.