Name: ____________________________

CS 0449: Lab 3 – Malloc and Free

Background

In class, we introduced C's facility for doing dynamic memory allocation, malloc(). **Malloc** is different from **new** in Java because malloc() takes a number of bytes rather than a data type. Another difference is the way the data is cleaned up after we're done with it. Unlike with garbage collection in Java, C requires you to explicitly free() all memory that has been allocated with malloc(). In this lab, we'll construct a simple linked list, traverse it, and then free it.

Building a Linked List

We saw in class that we can still easily implement linked lists in C, but rather than using a class to represent a node in the list, we can instead use a struct. For this lab, let's assume a simple node struct:

```
struct Node {
        int grade;
        struct Node *next;
}
```

Each node will represent one grade. Our first step is to populate a linked list with some grades. Write some code that reads in a bunch of integers from the console with scanf and stops when the user enters -1. For each integer entered, use malloc() to allocate a new Node and assign to grade the value entered. Build up a linked list

Traversing a Linked List

After the user has entered the data and you have constructed your linked list, traverse the list and find the average of the grades. Report this number to the user through a printf().

Cleaning Up a Linked List

When the data is no longer necessary and before your program terminates, it is always a good idea to make sure you have freed anything allocated by a call to malloc(). Your list is comprised of individual nodes that each was created by a call to malloc(). Thus to clean it up, you must call free() on each node of the linked list. However, we may not use the data in a malloc()ed space after it has been freed, and each node contains a pointer to find the next node. That means that we need to free the nodes from last element to first. Write some code that does this and then show the TA when you're done.