CSE31 : Lab #3 – Malloc

This lab contains two parts. To ensure you get full credit, make sure you read this lab carefully and follow the instructions precisely.

Overview

This lab will delve into malloc details and memory.

(Reference) Reading

K&R : 6.1-6.5

(Exercise) Memory

Download mem.c from the assignment page. Answer the following questions for **each** code statement in mem.c:

C1. What areas of memory (heap or stack) are affected by the statement ?

C2. Does any memory get allocated or freed?

C3. If C2 is true then where is this memory?

C4. Does the statement result in a memory leak?

(Exercise) Vector

Download Makefile, test.c, vector.c and vector.h from the assignment page. In vector.c, we provide you with a framework for implementing a variable-length array (in Java and C++, these variable-length arrays are known as the Vector class). This exercise is designed to help familiarize you with C structs and memory management in C.

Fill in the missing code (you only need to modify vector.c), and test it by running 'make test'. If the test breaks, try using gdb (or ddd) on the created executable (vector.test) or printf statements to find your bugs. If you have extra time, you can try adding more test cases to the code in test.c and explore how we've split up this one program into two different .c files.

Comments in the code describe how the functions should work. Look at the functions we've filled in to see how the data structures should be used.

For consistency, it is assumed that all entries in the vector are 0 unless set by the user. Keep this in mind as malloc() does not zero-out the memory it allocates.

What to hand in

When you are done with this lab assignment, you are ready to submit your work. Make sure you have done the following ***before*** you press Submit:

- Answers for each line of mem.c using **C1-C4**.
- Attach filled in vector.c
- List of collaborators