Homework 1 for CS202
Bradley Fallon
bfallon@pdx.edu
4/29/2019

UNIX DEBUG

The first step of testing and debugging I used was to simply compile the code with gdb and see what errors pop up. This will reveal the most basic issues such as syntax errors, typos, missing includes, major design flaws etc.

The next level of testing and debugging, after the program successfully compiles is to run with gdb and address any segmentation fault issues. I ran into this problem a lot on this assignment. I used a lot of dynamic memory. Using gdb I was able to put breakpoints and display variable values just before the errors occurred. This aided me in determining the cause of error. The two common causes were that I had gotten the address of some objects that were not dynamically allocated, and tried accessing these pointers later. The other common issue was that I needed to set all deleted memory pointers to NULL and also needed to check if pointers were NULL before attempting to delete.

The highest level of testing and debugging I used was memory analysis. I used valgrind to detect memory leaks. Using the command “valgrind --leak-check=yes ./a.out” I can see if there are memory leaks, and I can see where the memory was allocated. Being able to see where the memory is allocated is extremely helpful in a program of this size. I had to improvise many features of my design that were overlooked initially, and it is easy to forget about dynamically allocated memory. Using valgrind iteratively, I was eventually able to run my program with reported zero memory leak.