Homework 2 for CS202
Bradley Fallon
bfallon@pdx.edu
5/8/2019

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. This revealed issues with unused variables, typos, and with undefined methods. One issue I ran into that took me a while to understand was that my setter methods were used in my copy constructor, which would delete non-NULL pointers before setting new value, and my initialization list was missing, so the pointers were not set to NULL.

The next level of testing and debugging, after the program successfully compiles is to run with gdb and address any segmentation fault issues. I also use backtrace and display to understand what was the path of commands that lead to the error. One error I was running into was that my destructor was not working for my queue if I did not dequeue all packages before ending program. This was something related to iterative traversal for deletion. I rewrote the delete all to be recursive and fixed the issue.

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=full ./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. This helped track down issues with my destructors and copy constructors.Using valgrind iteratively, I was eventually able to run my program with reported zero memory leak.