# Test Plan

The test plan here is actually split up into unit tests for the individual Node, List, PriorityString, and PriorityQueue classes. Integration tests are then performed on Priority Queue.

## Memory

The first tests verify memory usage.

1. Operations on various types of Item declarations
   1. Stack Frame
   2. Heap
2. Tests to verify handling empty things
3. Tests to handle over and underflow
4. Tests to verify construction
5. Tests to verify destructor operations

Functional Tests

Testing of things like searching and inserting

# Lessons Learned

## Using your list

I did this assignment twice, the first solution can actually be seen here. But I wanted to talk about how I did it. At first I was trying to implement the sorted insert routine inside the ppqueue class. I had to move it into the list class because the head is declared as a private Node and I needed access to the node pointer to call node insert after I had found the appropriate location.

I encountered issues with inserting into an empty list with the insert methods that were provided. The lists constructor creates a List with the head node pointer set to null. The list\_head\_insert method of Node handle this by always inserting at head as we have discussed in class. The list\_insert method however accessesed previous\_ptr->link() without checking for null. This would cause a null de-reference exception when inserting using this method on an empty list.

## The power of basic\_node

This allows me to not template the iterator portions of my nodes! Cool! Imagine the fractions of kb this will save! It saves space because the iterator code will not be regenerated for each type that utilizes it.

I didn’t really get the idea from one place so much as several sources on the internet, but I implemented the code from scratch after reading. I also inspected the g++ source libraries on my computer so I guess that would be a source to cite as well.

## Iterators

I use

## Why the linked list sucks for a priority queue

# Using typedefs with Generics

As of c++ 2011 you can use

# Errors found

You are missing an #endif in the

# Future work

I would create a base class for priority that PriorityString inherits from that defines the getPriority method and member. This would allow more granular testing on priority form within the priority queue methods. I may also implement priority queue to operate on this class as opposed to a pure generic, and use generics in the underlying priority class.

# TODO list

Check for negative priorities