## Document

I felt like my other document was getting really long, and that I would have a lot of specific learning associated to this project so I made a new one.

I did not comment my code very much for this project. This was on purpose. It was becoming cluttered and I was changing types so frequently as I went that it was slowing me down greatly. I was also going to have so many “leanings” that it was going to be unreadable.

Instead I have decided to type of the things I learned here, and let the code speak for itself. That is not to say I have no comments, I made them where I felt further explanation was needed, but they are not to the level I know you prefer to have for your assignments and I wanted to note that I did this on purpose.

## 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.

## nullptr

There is a constant in the std call nullptr. It has implications when doing type checking. One of the more useful explanations I found was this post:

<http://stackoverflow.com/questions/13816385/what-are-the-advantages-of-using-nullptr>

It makes a lot of sense to include this. I come from a C background and we did not the std, so it is obvious to me that NULL is as a pre-compiler directive and hence will resolve as an int because I had to write the #DEFINE statement, but I don’t think that is the norm, and I think it is easy to forget, so introducing a type to handle null pointers was a smart move.

## Auto

C++11 adds support for an auto, which is similar to var in c#. It allows the compiler to deduce what type to use for your variable based on how you initialize it. IT’s not really “needed” but is one of those things that newer language are adding because it makes it “easier” to code.

“Specifies that the type of the variable that is being declared will be automatically deduced from its initializer. “

From <http://en.cppreference.com/w/cpp/language/auto>

I didn’t use it though, because it’s for c++11 only.

## Explicit constructors and conversion constructors

<http://en.cppreference.com/w/cpp/language/converting_constructor>