HashMap<int,string> h(11);  
if(h.empty()) {  
 h.put(24, "test");  
 pair<int, string> s = h.find(24);  
 cout << s.first << ", " << s.second << endl;  
 h.put(25, "test2");  
 h.put(27, "test3");  
 cout << h.find(25).first << "," << h.find(25).second << endl;  
 cout << h.size() << endl;  
 h.erase(25);  
 cout << h.size() << endl;  
 h.put(35, "test4");  
 cout << h.find(35).first << "," << h.find(35).second << endl;  
 h.erase(35);  
}  
BSTMap<int,string> z;  
if(z.empty())  
{  
 z.put(24, "test");  
 pair<int, string> m = z.find(24)->value;  
 cout << m.first << ", " << m.second << endl;  
 z.put(23, "test1");  
 z.put(25, "test2");  
 z.put(26, "test3");  
 cout << z.find(23)->value.first << "," << z.find(23)->value.second << endl;  
 cout << z.size() << endl;  
 z.erase(24);  
 cout << z.size() << endl;  
 cout << z.find(25)->value.first << "," << z.find(25)->value.second << endl;  
}

Expected Output:

24, test

25,test2

3

2

35,test4

24, test

23,test1

4

3

25,test2

These lines of code verify that all of the functions implemented work correctly. The items are input, found, and erased correctly. The actual implementation in the code show that the ordering and that the find() functions work more extensively due to having to find and put much more often. The actual implementation also verifies that the charDistribution functions work.

HashMap<string,charDistribution>\* m = new HashMap<string,charDistribution>(109);  
for (int i = 0; i+w < text.length()-1; i++)  
{  
 curr = text.substr(i,w);  
 pair<string,charDistribution> z = m->find(curr);  
 if(z.first.compare(curr)==0)  
 {  
 z.second.occurs(text[i+w]);  
 }  
 else  
 {  
 charDistribution n;  
 n.occurs(text[i+w]);  
 m->put(curr,n);  
 }  
 //cout << curr << endl;  
}

BSTMap<string,charDistribution>\* m = new BSTMap<string,charDistribution>();  
charDistribution n;  
n.occurs(text[w]);  
m->put(text.substr(0,w),n);  
for (int i = 1; i+w < text.length()-1; i++)  
{  
 curr = text.substr(i,w);  
 pair<string,charDistribution> z = m->find(curr)->value;  
 if(z.first.compare(curr)==0)  
 {  
 z.second.occurs(text[i+w]);  
 }  
 else  
 {  
 charDistribution n;  
 n.occurs(text[i+w]);  
 m->put(curr,n);  
 }  
 //cout << curr << endl;  
}

These implementations do not have output except in a text file named “TheRealCase.txt”. This file shows that all of the pieces work together to create a correct output. Included with this file is that text file showcasing the BSTMap’s work with a window of 15. After many tests, the HashMap gave similar results.