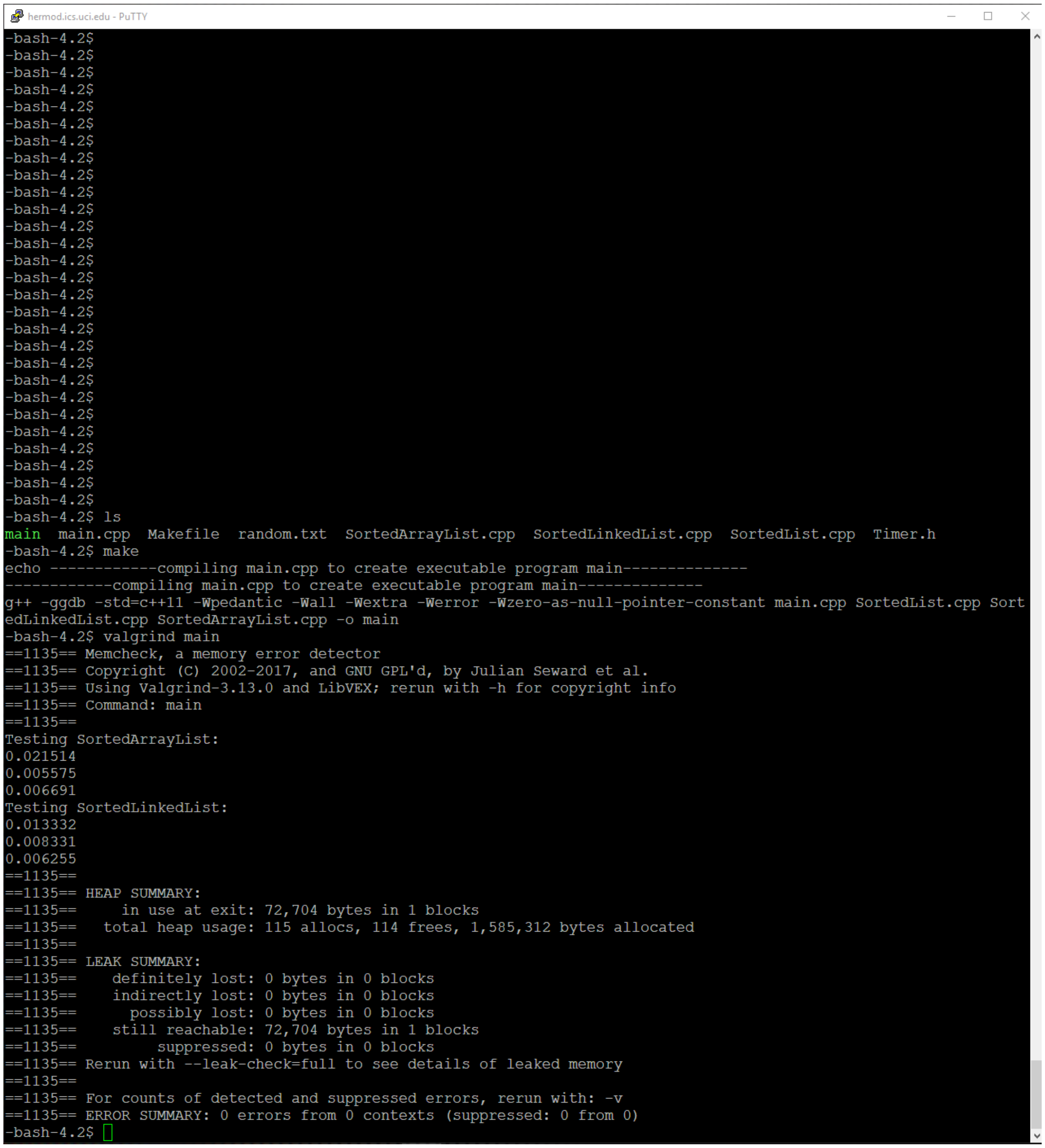
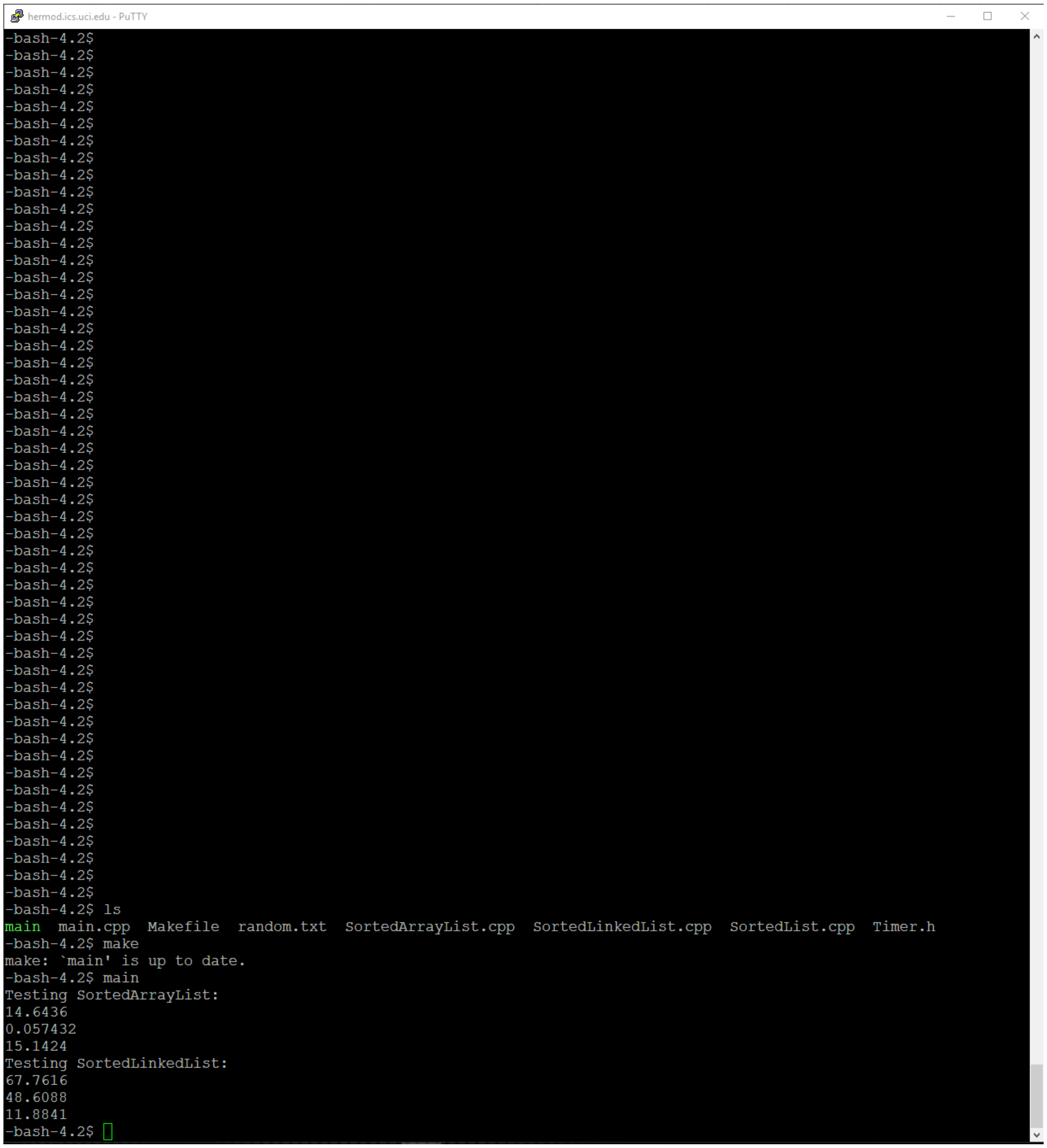
Homework 3

Beginning of Report



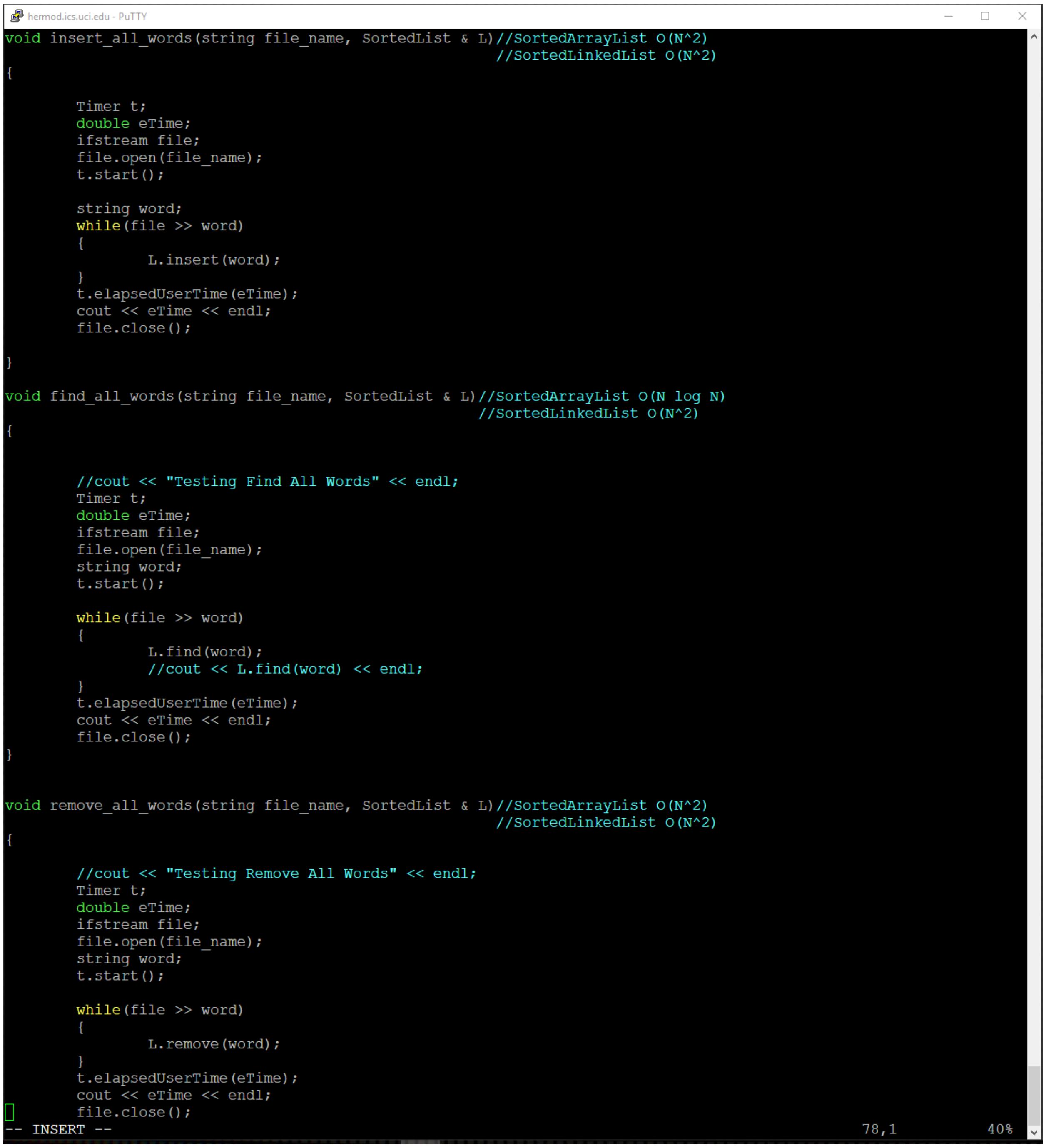
```
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$ ls
main  main.cpp  Makefile  random.txt  SortedArrayList.cpp  SortedLinkedList.cpp  SortedList.cpp  Timer.h
-bash-4.2$ make
echo ------------compiling main.cpp to create executable program main--------------
------------compiling main.cpp to create executable program main--------------
g++ -ggdb -std=c++11 -Wpedantic -Wall -Wextra -Werror -Wzero-as-null-pointer-constant main.cpp SortedList.cpp Sort
edLinkedList.cpp SortedArrayList.cpp -o main
-bash-4.2$ valgrind main
==1135== Memcheck, a memory error detector
==1135== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==1135== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==1135== Command: main
==1135==
Testing SortedArrayList:
0.021514
0.005575
0.006691
Testing SortedLinkedList:
0.013332
0.008331
0.006255
==1135==
==1135== HEAP SUMMARY:
==1135==     in use at exit: 72,704 bytes in 1 blocks
==1135==   total heap usage: 115 allocs, 114 frees, 1,585,312 bytes allocated
==1135==
==1135== LEAK SUMMARY:
==1135==    definitely lost: 0 bytes in 0 blocks
==1135==    indirectly lost: 0 bytes in 0 blocks
==1135==      possibly lost: 0 bytes in 0 blocks
==1135==    still reachable: 72,704 bytes in 1 blocks
==1135==         suppressed: 0 bytes in 0 blocks
==1135== Rerun with --leak-check=full to see details of leaked memory
==1135==
==1135== For counts of detected and suppressed errors, rerun with: -v
==1135== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
-bash-4.2$
```

Above is me compiling and running valgrind with a random.txt that has around 100 words.



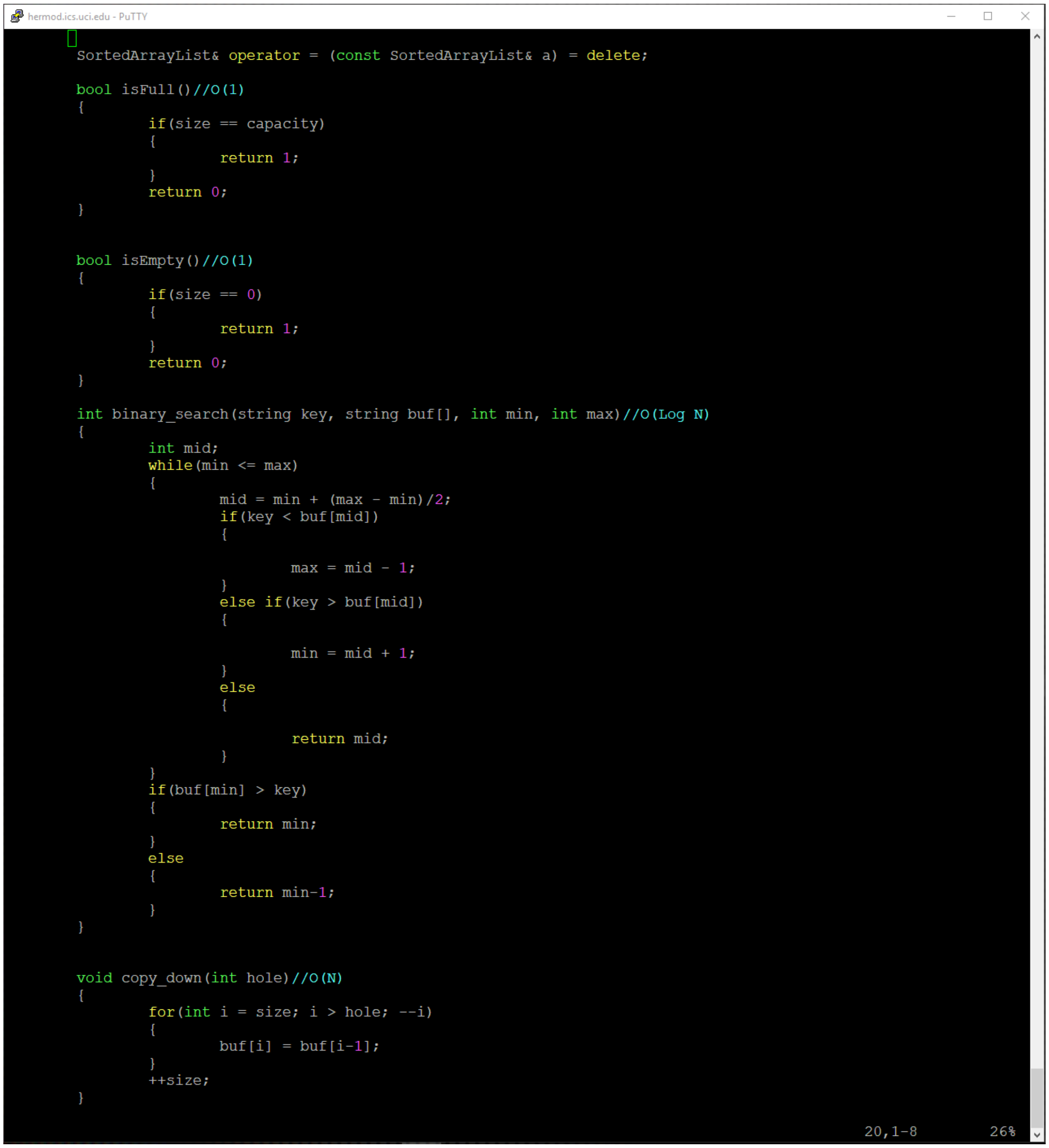
```
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$ ls
main  main.cpp  Makefile  random.txt  SortedArrayList.cpp  SortedLinkedList.cpp  SortedList.cpp  Timer.h
-bash-4.2$ make
make: `main' is up to date.
-bash-4.2$ main
Testing SortedArrayList:
14.6436
0.057432
15.1424
Testing SortedLinkedList:
67.7616
48.6088
11.8841
-bash-4.2$
```

Above is a regular run of the program using a random.txt file that has 45,000 words.

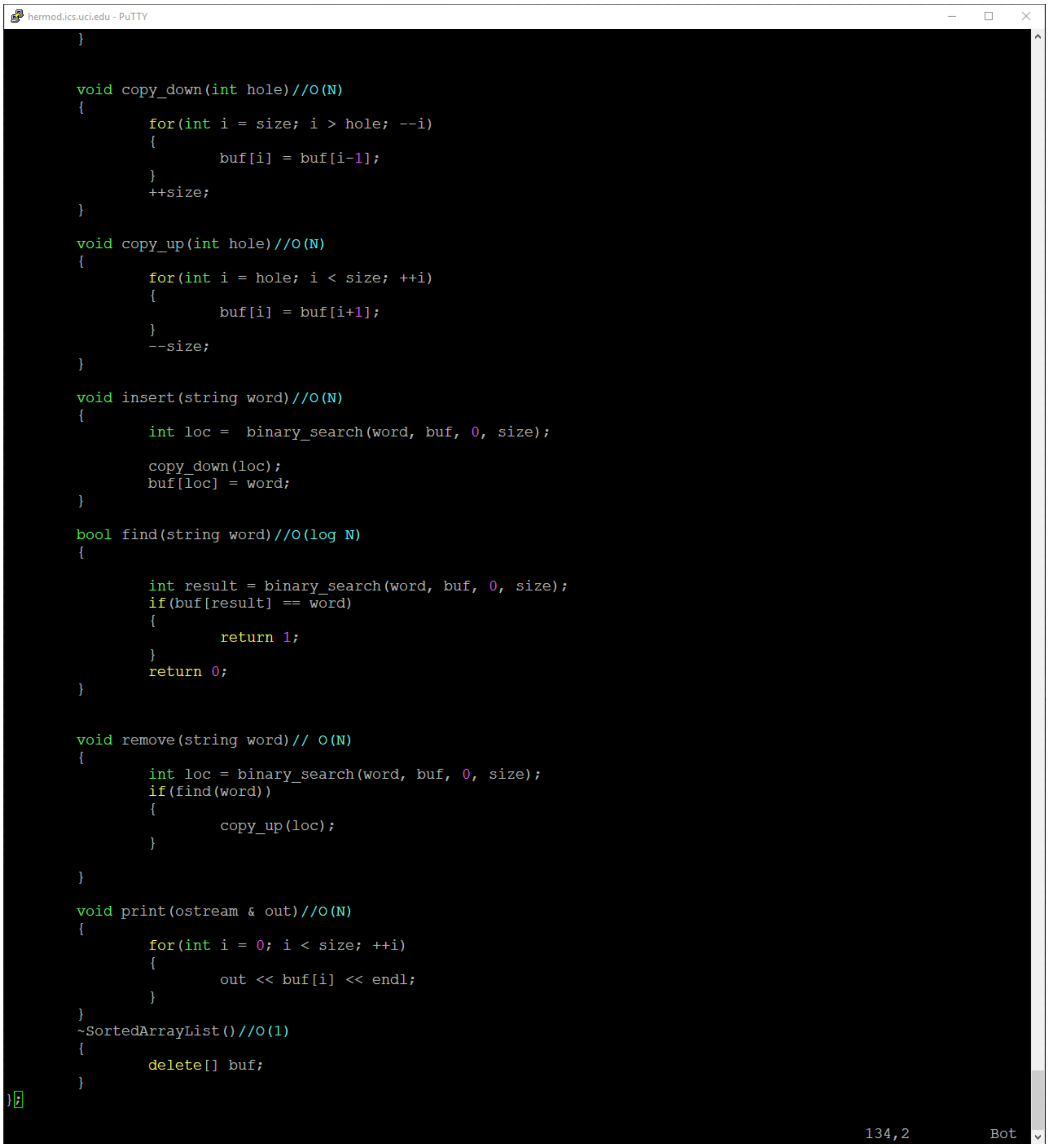


```
void insert_all_words(string file_name, SortedList & L)//SortedArrayList O(N^2)
                                                       //SortedLinkedList O(N^2)
{

        Timer t;
        double eTime;
        ifstream file;
        file.open(file_name);
        t.start();

        string word;
        while(file >> word)
        {
                L.insert(word);
        }
        t.elapsedUserTime(eTime);
        cout << eTime << endl;
        file.close();

}

void find_all_words(string file_name, SortedList & L)//SortedArrayList O(N log N)
                                                     //SortedLinkedList O(N^2)
{


        //cout << "Testing Find All Words" << endl;
        Timer t;
        double eTime;
        ifstream file;
        file.open(file_name);
        string word;
        t.start();

        while(file >> word)
        {
                L.find(word);
                //cout << L.find(word) << endl;
        }
        t.elapsedUserTime(eTime);
        cout << eTime << endl;
        file.close();
}


void remove_all_words(string file_name, SortedList & L)//SortedArrayList O(N^2)
                                                       //SortedLinkedList O(N^2)
{

        //cout << "Testing Remove All Words" << endl;
        Timer t;
        double eTime;
        ifstream file;
        file.open(file_name);
        string word;
        t.start();

        while(file >> word)
        {
                L.remove(word);
        }
        t.elapsedUserTime(eTime);
        cout << eTime << endl;
        file.close();
-- INSERT --                                                                      78,1          40%
```

Above is the O(N) of the test functions that use the methods from my SortedArrayList class and my SortedLinkedList class.



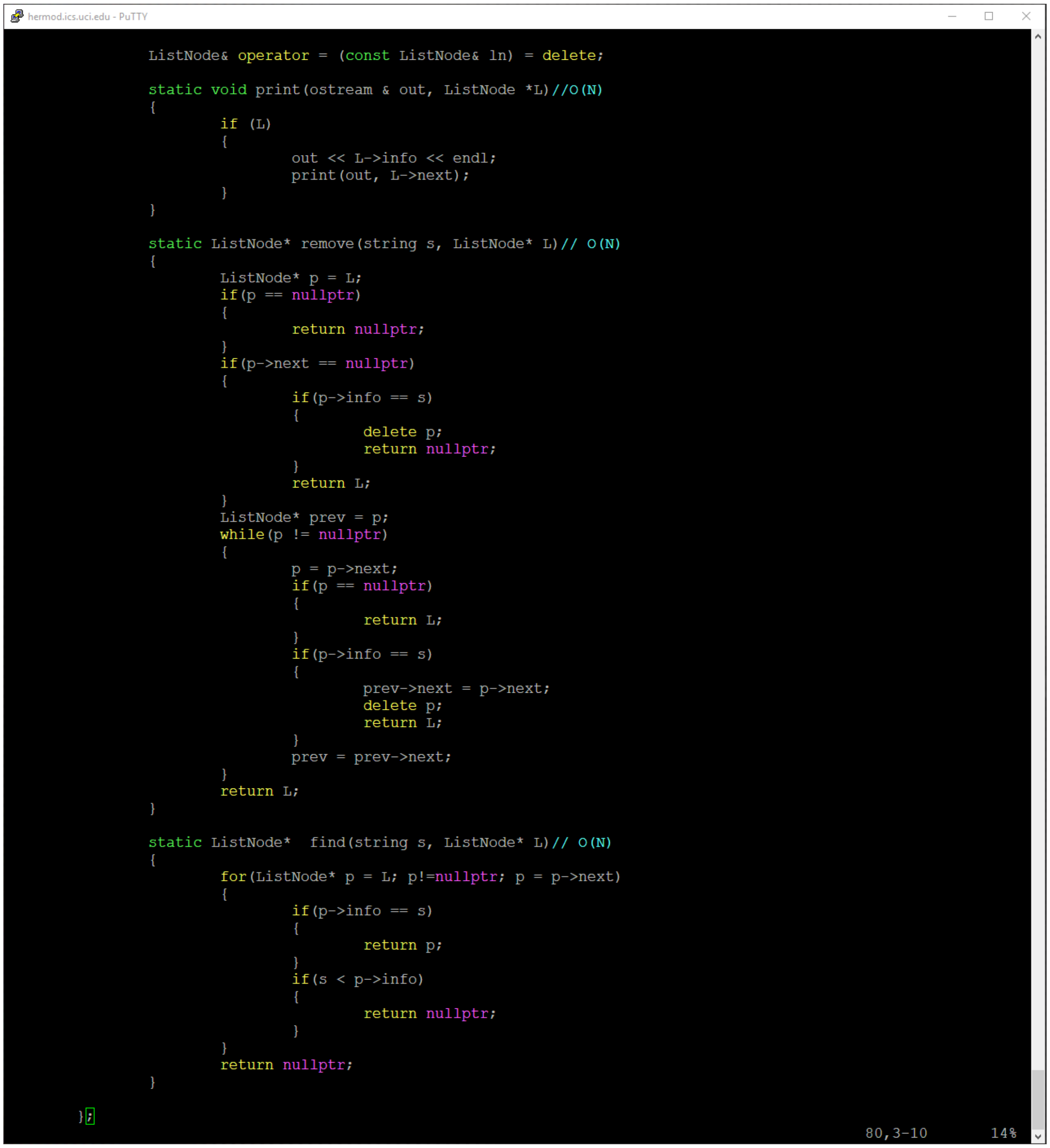
```
        SortedArrayList& operator = (const SortedArrayList& a) = delete;

        bool isFull()//O(1)
        {
                if(size == capacity)
                {
                        return 1;
                }
                return 0;
        }


        bool isEmpty()//O(1)
        {
                if(size == 0)
                {
                        return 1;
                }
                return 0;
        }

        int binary_search(string key, string buf[], int min, int max)//O(Log N)
        {
                int mid;
                while(min <= max)
                {
                        mid = min + (max - min)/2;
                        if(key < buf[mid])
                        {

                                max = mid - 1;
                        }
                        else if(key > buf[mid])
                        {

                                min = mid + 1;
                        }
                        else
                        {

                                return mid;
                        }
                }
                if(buf[min] > key)
                {
                        return min;
                }
                else
                {
                        return min-1;
                }
        }


        void copy_down(int hole)//O(N)
        {
                for(int i = size; i > hole; --i)
                {
                        buf[i] = buf[i-1];
                }
                ++size;
        }

                                                                                              20,1-8        26%
```



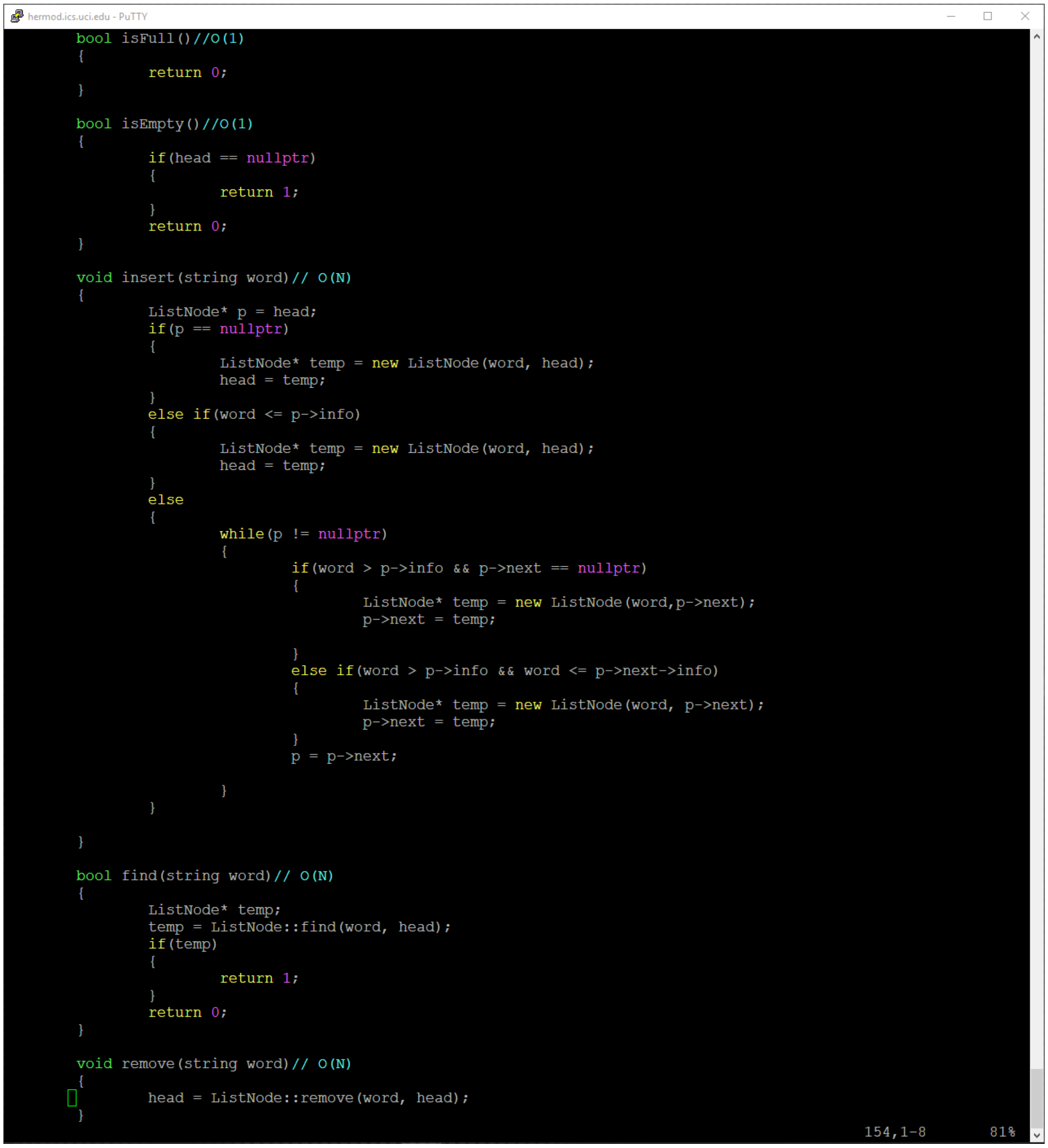
hermod.ics.uci.edu - PuTTY

```
        }


        void copy_down(int hole)//O(N)
        {
                for(int i = size; i > hole; --i)
                {
                        buf[i] = buf[i-1];
                }
                ++size;
        }

        void copy_up(int hole)//O(N)
        {
                for(int i = hole; i < size; ++i)
                {
                        buf[i] = buf[i+1];
                }
                --size;
        }

        void insert(string word)//O(N)
        {
                int loc =  binary_search(word, buf, 0, size);

                copy_down(loc);
                buf[loc] = word;
        }

        bool find(string word)//O(log N)
        {

                int result = binary_search(word, buf, 0, size);
                if(buf[result] == word)
                {
                        return 1;
                }
                return 0;
        }


        void remove(string word)// O(N)
        {
                int loc = binary_search(word, buf, 0, size);
                if(find(word))
                {
                        copy_up(loc);
                }

        }

        void print(ostream & out)//O(N)
        {
                for(int i = 0; i < size; ++i)
                {
                        out << buf[i] << endl;
                }
        }
        ~SortedArrayList()//O(1)
        {
                delete[] buf;
        }
};
                                                                                                                    134,2          Bot
```

Above is the O(N) of all of my methods in my SortedArrayList class.



```
        ListNode& operator = (const ListNode& ln) = delete;

        static void print(ostream & out, ListNode *L)//O(N)
        {
                if (L)
                {
                        out << L->info << endl;
                        print(out, L->next);
                }
        }

        static ListNode* remove(string s, ListNode* L)// O(N)
        {
                ListNode* p = L;
                if(p == nullptr)
                {
                        return nullptr;
                }
                if(p->next == nullptr)
                {
                        if(p->info == s)
                        {
                                delete p;
                                return nullptr;
                        }
                        return L;
                }
                ListNode* prev = p;
                while(p != nullptr)
                {
                        p = p->next;
                        if(p == nullptr)
                        {
                                return L;
                        }
                        if(p->info == s)
                        {
                                prev->next = p->next;
                                delete p;
                                return L;
                        }
                        prev = prev->next;
                }
                return L;
        }

        static ListNode*  find(string s, ListNode* L)// O(N)
        {
                for(ListNode* p = L; p!=nullptr; p = p->next)
                {
                        if(p->info == s)
                        {
                                return p;
                        }
                        if(s < p->info)
                        {
                                return nullptr;
                        }
                }
                return nullptr;
        }

};
                                                                80,3-10       14%
```



hermod.ics.uci.edu - PuTTY

```
        bool isFull()//O(1)
        {
                return 0;
        }

        bool isEmpty()//O(1)
        {
                if(head == nullptr)
                {
                        return 1;
                }
                return 0;
        }

        void insert(string word)// O(N)
        {
                ListNode* p = head;
                if(p == nullptr)
                {
                        ListNode* temp = new ListNode(word, head);
                        head = temp;
                }
                else if(word <= p->info)
                {
                        ListNode* temp = new ListNode(word, head);
                        head = temp;
                }
                else
                {
                        while(p != nullptr)
                        {
                                if(word > p->info && p->next == nullptr)
                                {
                                        ListNode* temp = new ListNode(word,p->next);
                                        p->next = temp;

                                }
                                else if(word > p->info && word <= p->next->info)
                                {
                                        ListNode* temp = new ListNode(word, p->next);
                                        p->next = temp;
                                }
                                p = p->next;

                        }
                }

        }

        bool find(string word)// O(N)
        {
                ListNode* temp;
                temp = ListNode::find(word, head);
                if(temp)
                {
                        return 1;
                }
                return 0;
        }

        void remove(string word)// O(N)
        {
                head = ListNode::remove(word, head);
        }
                                                                    154,1-8       81%
```

Above is the O(N) of my methods and static methods in my SortedLinkedList class.

End of Report