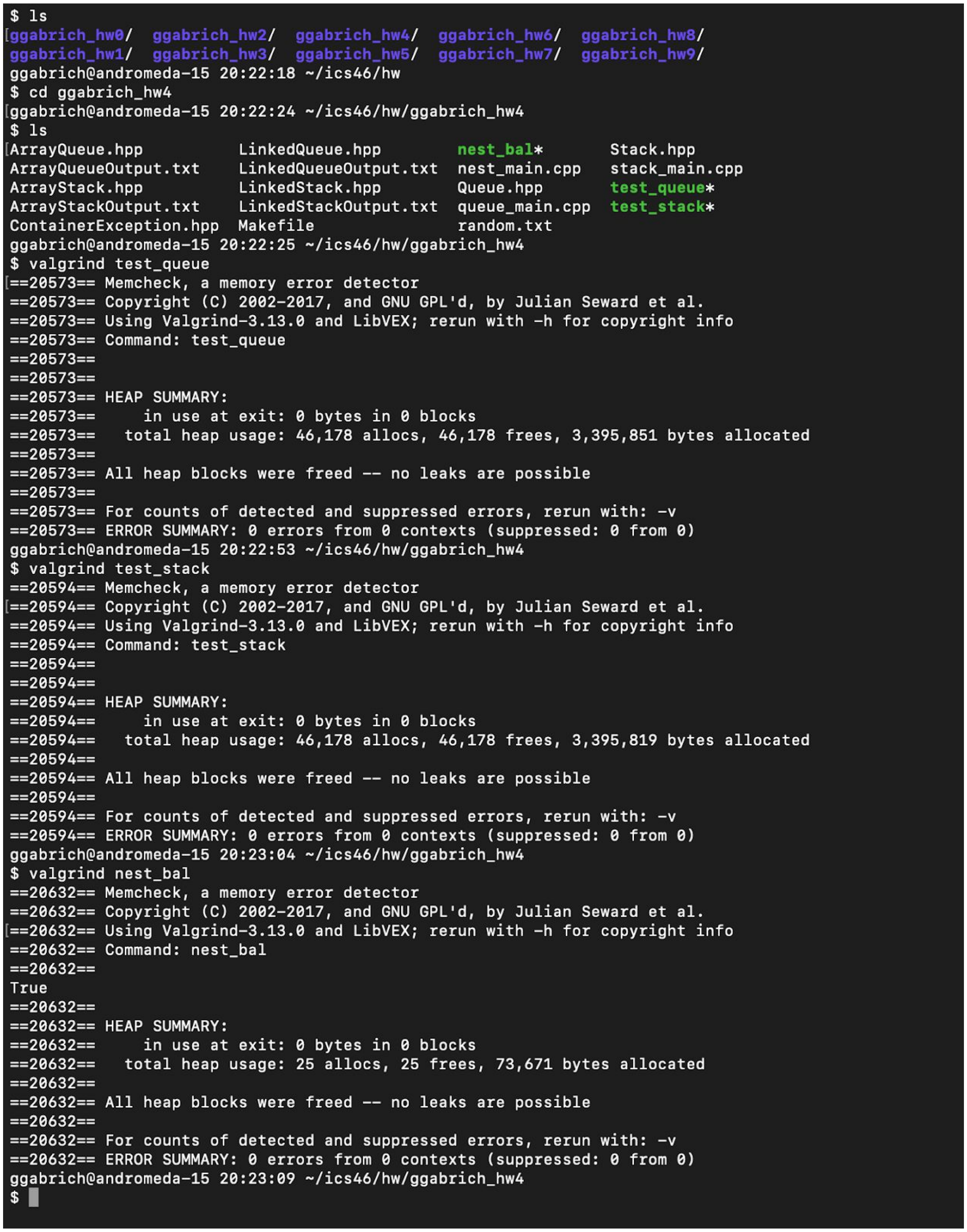
George Gabricht
ggabrich - 56735102



```
$ ls
[ggabrich_hw0/  ggabrich_hw2/  ggabrich_hw4/  ggabrich_hw6/  ggabrich_hw8/
 ggabrich_hw1/  ggabrich_hw3/  ggabrich_hw5/  ggabrich_hw7/  ggabrich_hw9/
 ggabrich@andromeda-15 20:22:18 ~/ics46/hw
 $ cd ggabrich_hw4
[ggabrich@andromeda-15 20:22:24 ~/ics46/hw/ggabrich_hw4
 $ ls
[ArrayQueue.hpp          LinkedQueue.hpp        nest_bal*      Stack.hpp
 ArrayQueueOutput.txt    LinkedQueueOutput.txt  nest_main.cpp  stack_main.cpp
 ArrayStack.hpp          LinkedStack.hpp        Queue.hpp      test_queue*
 ArrayStackOutput.txt    LinkedStackOutput.txt  queue_main.cpp test_stack*
 ContainerException.hpp  Makefile               random.txt
 ggabrich@andromeda-15 20:22:25 ~/ics46/hw/ggabrich_hw4
 $ valgrind test_queue
[==20573== Memcheck, a memory error detector
 ==20573== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
 ==20573== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
 ==20573== Command: test_queue
 ==20573==
 ==20573==
 ==20573== HEAP SUMMARY:
 ==20573==     in use at exit: 0 bytes in 0 blocks
 ==20573==   total heap usage: 46,178 allocs, 46,178 frees, 3,395,851 bytes allocated
 ==20573==
 ==20573== All heap blocks were freed -- no leaks are possible
 ==20573==
 ==20573== For counts of detected and suppressed errors, rerun with: -v
 ==20573== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
 ggabrich@andromeda-15 20:22:53 ~/ics46/hw/ggabrich_hw4
 $ valgrind test_stack
 ==20594== Memcheck, a memory error detector
[==20594== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
 ==20594== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
 ==20594== Command: test_stack
 ==20594==
 ==20594==
 ==20594== HEAP SUMMARY:
 ==20594==     in use at exit: 0 bytes in 0 blocks
 ==20594==   total heap usage: 46,178 allocs, 46,178 frees, 3,395,819 bytes allocated
 ==20594==
 ==20594== All heap blocks were freed -- no leaks are possible
 ==20594==
 ==20594== For counts of detected and suppressed errors, rerun with: -v
 ==20594== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
 ggabrich@andromeda-15 20:23:04 ~/ics46/hw/ggabrich_hw4
 $ valgrind nest_bal
 ==20632== Memcheck, a memory error detector
 ==20632== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
[==20632== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
 ==20632== Command: nest_bal
 ==20632==
 True
 ==20632==
 ==20632== HEAP SUMMARY:
 ==20632==     in use at exit: 0 bytes in 0 blocks
 ==20632==   total heap usage: 25 allocs, 25 frees, 73,671 bytes allocated
 ==20632==
 ==20632== All heap blocks were freed -- no leaks are possible
 ==20632==
 ==20632== For counts of detected and suppressed errors, rerun with: -v
 ==20632== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
 ggabrich@andromeda-15 20:23:09 ~/ics46/hw/ggabrich_hw4
 $
```

George Gabricht
ggabrich - 56735102

```
bool isBalanced(string & word) { // O(N)
    string sub;
    LinkedStack stk;
    int len = word.length();
    try {
        for (int ndx = 0; ndx < len; ndx++) {
            sub = word.substr(ndx, 1);
            // cout << "Sub: " << sub << endl;
            switch(sub[0]) {
                case '(': {
                    stk.push(sub);
                    break;
                }
                case '{': {
                    stk.push(sub);
                    break;
                }
                case '<': {
                    stk.push(sub);
                    break;
                }
                case '[': {
                    stk.push(sub);
                    break;
                }
                case ')': {
                    if (stk.top() == "(") {
                        stk.pop();
                    } else {
                        return false;
                    }
                    break;
                }
                case '}': {
                    if (stk.top() == "{") {
                        stk.pop();
                    } else {
                        return false;
                    }
                    break;
                }
                case '>': {
                    if (stk.top() == "<") {
                        stk.pop();
                    } else {
                        return false;
                    }
                    break;
                }
                case ']': {
                    if (stk.top() == "[") {
                        stk.pop();
                    } else {
                        return false;
                    }
                    break;
                }
                default:
                    return false;
            }
            //if (!stk.isEmpty()) {
            //      cout << "Stk.top(): " << stk.top() << endl;
            //}
        }
        return stk.isEmpty();
    } catch (ContainerUnderflow) {
        return false;
    }
}
```

LINKEDQUEUE

```
void fillAll(Queue * que, ifstream & instream) { // O(N)
    string curLine;
    while (getline(instream, curLine)) {
        que->enq(curLine);
    }
}

        static QueueNode * enq(string & word, QueueNode * nod) { // O(1)
            nod->next = new QueueNode(word, nullptr);
            return nod->next;
        }

        static string deq(QueueNode * nod) { // O(1)
            string result = nod->value;
            delete nod;
            return result;
        }

        static string front(QueueNode * nod) { // O(1)
            return nod->value;
        }

    virtual void enq(string & word) override { // O(1)
        if (isEmpty()) {
            head = tail = new QueueNode(word, nullptr);
```

George Gabricht
ggabrich - 56735102

```
        } else {
            tail = QueueNode::enq(word, tail);
        }
    }

    virtual string deq() override { // O(1)
        if (isEmpty()) {
            char msg[] = "Error: Dequeue on Empty Queue";
            throw ContainerUnderflow(msg);
        } else if (head == tail) {
            tail = nullptr;
        }
        QueueNode * temp = head;
        head = head->next;
        return QueueNode::deq(temp);
    }

    virtual string front() override { // O(1)
        if (isEmpty()) {
            char msg[] = "Error: Front on Empty Queue";
            throw ContainerUnderflow(msg);
        }
        return QueueNode::front(head);
    }

    virtual bool isEmpty() override { // O(1)
        return head == nullptr;
    }

    virtual bool isFull() override { // O(1)
        return false;
    }


void emptyAll(Queue * que, ofstream & outstream) { // O(N)
    while (!que->isEmpty()) {
        outstream << que->deq() << endl;
    }
}
```

LINKEDSTACK

```
void fillAll(Stack * stk, ifstream & instream) { // O(N)
    string curLine;
    while (getline(instream, curLine)) {
        stk->push(curLine);
    }
}


        static StackNode * push(string & word, StackNode * nod) { // O(1)
```

George Gabricht
ggabrich - 56735102

```
            return new StackNode(word, nod);
        }

        static string pop(StackNode * nod) { // O(1)
            string result = nod->value;
            delete nod;
            return result;
        }

        static string top(StackNode * nod) { // O(1)
            return nod->value;
        }

    virtual void push(string & word) override { // O(1)
        head = StackNode::push(word, head);
    }

    virtual string pop() override { // O(1)
        if (isEmpty()) {
            char msg[] = "Error: Pop on Empty Stack.";
            throw ContainerUnderflow(msg);
        }
        StackNode * temp = head;
        head = head->next;
        return StackNode::pop(temp);
    }

    virtual string top() override { // O(1)
        if (isEmpty()) {
            char msg[] = "Error: Top on Empty Stack.";
            throw ContainerUnderflow(msg);
        }
        return StackNode::top(head);
    }

    virtual bool isEmpty() override { // O(1)
        return head == nullptr;
    }

    virtual bool isFull() override { // O(1)
        return false;
    }


void emptyAll(Stack * stk, ofstream & outstream) { // O(N)
    while (!stk->isEmpty()) {
        outstream << stk->pop() << endl;
    }
}
```