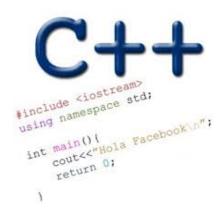
LINKED LISTS (CONTD) RULE OF THREE DEALING WITH MEMORY ERRORS MORE ON OPERATOR OVERLOADING

Problem Solving with Computers-II





Memory Errors

Memory Leak: Program does not free memory allocated on the heap.

Segmentation Fault: Code tries to access an invalid memory location

RULE OF THREE

If a class overload one (or more) of the following methods, it should overload all three methods:

- 1. Destructor
- 2. Copy constructor
- 3. Copy assignment

The questions we ask are:

- 1. What is the behavior of these defaults?
- 2. What is the desired behavior?
- 3. How should we over-ride these methods?

Behavior of default destructor

```
void test_append_0(){
     vector<int> v exp = {1};
     LinkedList 11;
     11.append(1);
     vector<int> v act = ll.vectorize();
     TESTEQ(v_exp, v_act, "test 0");
Assume:
destructor: default
copy constructor: default
copy assignment: default
```

What is the output?

- A. Compiler error
- B. Memory leak
- C. Segmentation fault
- D. Test fails
- E. None of the above

Why do we need to write a destructor for LinkedList?

- A. To free LinkedList objects
- B. To free Nodes in a LinkedList
- C. Both A and B
- D. None of the above

Behavior of default copy constructor

```
void test copy constructor(){
   LinkedList 11;
   11.append(1);
   11.append(2);
  LinkedList 12(11);
   TESTEQ(11, 12, "test copy constructor");
                                     What is the output?
                                     A. Compiler error
Assume:
                                     B. Memory leak
destructor: overloaded
                                     C. Segmentation fault
copy constructor: default
                                     D. Test fails
```

E. None of the above

copy assignment: default

Behavior of default copy assignment

```
void test_copy assignment(){
  LinkedList 11;
   11.append(1);
   11.append(2);
  LinkedList 12;
   12 = 11;
  TESTEQ(11, 12, "test copy assignment");
Assume:
destructor: overloaded
copy constructor: overloaded
copy assignment: default
```

What is the output?

- A. Compiler error
- B. Memory leak
- C. Segmentation fault
- D. Test fails
- E. None of the above

Write another test case for the copy assignment

```
void test_copy_assignment_2(){
```

Overloading Binary Comparison Operators

We would like to be able to compare two objects of the class using the following operators

==

!=

and possibly others

Last class: overloaded == for LinkedList

Overloading Binary Arithmetic Operators

We would like to be able to add two points as follows

```
LinkedList 11, 12;

//append nodes to 11 and 12;

LinkedList 13 = 11 + 12;
```

Overloading input/output stream

Wouldn't it be convenient if we could do this:

```
LinkedList list;
cout<<li>t; //prints all the elements of list
```

Next time

• Recursion + PA01