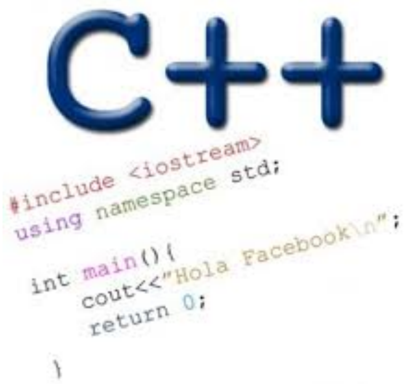
LINKED LISTS (CONTD)
RULE OF THREE
DEALING WITH MEMORY ERRORS
MORE ON OPERATOR OVERLOADING

Problem Solving with Computers-II



```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```

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 ll;
    ll.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 l1;
   l1.append(1);
   l1.append(2);
   LinkedList l2(l1);
   TESTEQ(l1, l2, "test copy constructor”);
}
```

Assume:

destructor: overloaded

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

Behavior of default copy assignment

```
void test_copy_assignment(){
   LinkedList l1;
   l1.append(1);
   l1.append(2);
   LinkedList l2;
   l2 = l1;
   TESTEQ(l1, l2, "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

Overloading Binary Arithmetic Operators

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

```
LinkedList l1, l2;

//append nodes to l1 and l2;

LinkedList l3 = l1 + l2 ;
```

Overloading input/output stream

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

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

GDB: GNU Debugger

- To use gdb, compile with the -g flag
- Setting breakpoints (b)
- Running programs that take arguments within gdb (r arguments)
- Continue execution until breakpoint is reached (c)
- Stepping into functions with step (s)
- Stepping over functions with next (n)
- Re-running a program (r)
- Examining local variables (info locals)
- Printing the value of variables with print (p)
- Quitting gdb (q)
- Debugging segfaults with backtrace (bt)

* Refer to the gdb cheat sheet: http://darkdust.net/files/GDB%20Cheat%20Sheet.pdf

Next time

- Recursion + PA01