Notes CS328

As we all know, pointers are useful & necessary tools in C++ for effective programming. But, the built-in pointers are too powerful & too weak. They allow the programmers too much freedom with too little oversight.

Aggregation is the combining of objects within other objects. When aggregating, you must be aware of the issue of memory management. If a pointer is contained, then who the life of the object that is being pointed to?

If a built-in pointer is being contained in an aggregate, we have referential aggregation. Because of the behavior of the default assignment operator & copy constructor, when an aggregate is copied, the pointer is copied but not what is pointed to (shallow copy) & so we have 2 pointers pointing to the same object. This leads to referential aggravation. Leaks are caused in all sorts of ways, & you can try as yyou might to write adequate copy constructors & assignment operators to prevent this but even if so, you can still leak memory.

Programmers defined “smart pointers” act to interrupt indirection so as to perform “house cleaning” duties & their reaction to that indirection. We write pointer classes to establish a relationship between pointer & object. We look at 2 types of pointer classes:

---REFERENCE CopiedBuiltInPtr class p 426----

The benefits:

* A one –to-one relationship between pointer object & object pointed to to
* Automatic copy made when the c++ generated copy constructor is built sonce it will automatically call the C.C. of its member
* The delete will properly delete heap objects
* Assignment operator will new heap object
* Have the overloaded pointer derenferencing operator, \*

----REFERENCE CopiedObjPtr class p 430----

Note: the CBPtr is for built-in ptrs & should not have the member access operator (->). The COPtr class does. It should return either a built-in pointer to a class object or an object of a ptr class typer that has a na overloaded -> operator (a built-in ptr to built-in object can’t be returned).

Example of a leak:

---REFERENCE Leaker class p 423-424----

How to fix: (not in text)

class NonLeaker

{

public:

NonLeaker(int size): p(new int(0)), arr(size) {}

private:

CBPtr<int> p;

Array<int> arr;

};