Ryan Hoffman
CS202 Project X Documentation
4/6/2019
Compile Command: make all

cout << endl << "Testing SmartPtr Default ctor" << endl;
SmartPtr sp1; // Default-ctor
sp1->setIntVal(1);
sp1->setDoubleVal(0.25);
cout << "Dereference Smart Pointer 1: " << *sp1 << endl;
-Creates a smart pointer object using the default constructor.
-Uses -> operator overload to set the int and double values of the smart pointer object.
-Prints the smart pointer object by using the * operator overload and the << operator overload.

cout << endl << "Testing SmartPtr Copy ctor" << endl;
SmartPtr sp2 = sp1; // Copy-initalization (Copy-ctor)
sp2->setIntVal(2);
sp2->setDoubleVal(0.5);
cout << "Dereference Smart Pointer 1: " << *sp1 << endl;
cout << "Dereference Smart Pointer 2: " << *sp2 << endl;
-Creates a new smart pointer object using the copy constructor and the first smart pointer object as a parameter.
-Sets the int and double values of sp2 using the -> operator overload.
-Prints the data for the first and second smart pointers using the * operator overload and the << operator overload.
-Since sp2 is pointed at sp1, they will both print the same thing.

cout << endl << "Testing SmartPtr Assignment operator" << endl;
SmartPtr sp3;
sp3 = sp1; // Assignment operator
sp3->setIntVal(4);
sp3->setDoubleVal(0.0);
cout << "Dereference Smart Pointer 1: " << *sp1 << endl;
cout << "Dereference Smart Pointer 2: " << *sp2 << endl;
cout << "Dereference Smart Pointer 3: " << *sp3 << endl;
-Creates a third smart pointer object using the default constructor.
-Uses the -> operator overload to set the integer value of sp3 to 4 and the double value of sp3 to 0.0.

-The * operator overload and << operator overload are used to print the data of all 3 smart pointers. Since they are all pointed at sp1, all of them will print the same thing.

cout << endl << "Testing SmartPtr Parametrized ctor with NULLdata" << endl;
SmartPtr spNull(NULL); // NULL-data initialization
cout << endl << "Testing SmartPtr Copy ctor with NULLdata SmartPtr" << endl;
SmartPtr spNull_cpy(spNull); // NULL-data copy constructor
-Creates a new smart pointer using the parameterized constructor and passing NULL in as a parameter.
-Creates a new smart pointer object using the copy constructor and passing in the previous NULL smart object as a parameter. All value sof spNull_copy will be equal to all values of spNull.

cout << endl << "Testing SmartPtr Assignment with NULLdata SmartPtr" << endl;
SmartPtr spNull_assign;
spNull_assign = spNull; // NULL-data assign
-Creates a new smart pointer object using the default constructor.
-Sets all the values of spNull_assign to all values of spNull.

cout << endl << "End-of-Scope, Destructors called in reverse order of SmartPtr creation\n(spNull_assign, spNull_cpy, spNull, sp3, sp2, sp1): " << endl;
-Since the program is over, the destructors are called for all smart pointer objects.

The purpose of this program was to get more practice with allocating memory inside classes. The design was pretty self explanatory, I followed the directions for each function within the project guidelines. If I was going to be allocating new memory I needed to first check if any memory was already allocated so I wouldn't get an error. The only problem I ran into was that I accidently deleted my whole project once I finished and I couldn't get it back so I had to restart. The only other thing I had to fix was that when I was incrementing the values of the pointer I would do *m_refcount++ when I should have been doing (*m_refcount)++. If I had more time I would add try/catch/throw to my problem to check memory allocation.