Purpose
The purpose of this program is to build a limited SmartPtr type that wraps a given data type and manages memory on the heap.

Design
I followed the specifications. I wrote two utility functions: One for decreasing the ref count and deallocating and one for creating a smart pointer from a given pointer. These functions helped reduce code duplication throughout my project.

Problems/Changes
I could not find any new test cases to add to the given one. I also did not implement any checks to make sure allocations succeeded. If I had more time, defining an “invalid” state of the pointer that could be tested for would be ideal in order to make errors obvious.

Program Output
```
Testing SmartPtr Default ctor ← Test Output
SmartPtr Default Constructor for new allocation, RefCount=1 ← Default Constructor
Dereference Smart Pointer 1: {1,0.25} ← Default inner values

Testing SmartPtr Copy ctor ← Test Output
SmartPtr Copy Constructor, RefCount=2 ← Copy Constructor
Dereference Smart Pointer 1: {2,0.5} ← Inner value of first pointer
Dereference Smart Pointer 2: {2,0.5} ← Inner value of second pointer

Testing SmartPtr Assignment operator ← Test Output
SmartPtr Default Constructor for new allocation, RefCount=1 ← Default Constructor
SmartPtr Copy Assignment, RefCount=3 ← SmartPtr Assignment
Dereference Smart Pointer 1: {4,0} ← Value of first smartptr
Dereference Smart Pointer 2: {4,0} ← Value of second smartptr
Dereference Smart Pointer 3: {4,0} ← Value of third Smartptr

Testing SmartPtr Parametrized ctor with NULLdata ← Test Output
SmartPtr Parametrized Constructor from data pointer, RefCount=0 ← Parameterized
Constructor

Testing SmartPtr Copy ctor with NULLdata SmartPtr ← Test Output
SmartPtr Copy Constructor, RefCount=0 ← Copy Constructor of NULL SmartPtr

Testing SmartPtr Assignment with NULLdata SmartPtr ← Test Output
SmartPtr Default Constructor for new allocation, RefCount=1 ← Default Constructor
SmartPtr Copy Assignment, RefCount=0 ← Copy Assign of NULL SmartPtr

End-of-Scope, Destructors called in reverse order of SmartPtr creation ← Test Output
```

```
(spNull_assign, spNull_cpy, spNull, sp3, sp2, sp1):
SmartPtr Destrcutor, RefCount=0 ← Destructor
SmartPtr Destrcutor, RefCount=0 ← Destructor
SmartPtr Destrcutor, RefCount=0 ← Destructor
SmartPtr Destrcutor, RefCount=2 ← Destructor, no deallocation
SmartPtr Destrcutor, RefCount=1 ← Destructor, no deallocation
SmartPtr Destrcutor, RefCount=0 ← Destructor, deallocation
```