# Dynamic Memory Management Functions, Reading

Since our arrayed implementation uses a dynamic array, we'll need the standard three memory management functions (to avoid memory leaks and to support making object copies). Here are the prototypes:

```
~AssociativeArray();  // destructor prototype
AssociativeArray(const AssociativeArray<K,V>&); // copy constructor prototype
AssociativeArray<K,V>& operator=(const AssociativeArray<K,V>&); // assignment operator prototype
```

They are the same as DynamicArray's except for the addition of **typename K** for the key.

The coding in their function templates are the same as for DynamicArray with these exceptions:

1. the array data member is **Node* data** instead of **V* values**, and
2. the private data member "siz" needs to be copied in the copy constructor and the assignment operator, exactly as "cap" is -- **siz=original.siz;**.

Writing these functions is left as an exercise for the student.