Lab 1: BigFive Class

The **Big Three** can be expanded to the **Big Five**, which include a **constructor** and an **overloaded assignment operator** that "**steal**" all the data from the parameter object.

We know that the job of a **constructor** is to create a new object, and we know that the job of a **copy constructor** is to create a new object that is an exact copy of the object passed as a parameter. The **move constructor** also creates a new object, but instead of allocating new memory (as the copy constructor does), it "steals" any dynamic data that belongs to the **parameter object**. Since the **parameter object** will not be used any longer, it is good practice to null its member pointers and reset any additional member variables to default values.

The **move assignment operator** works in the same manner by "**stealing**" the data from the **parameter object** (note that, as with the overloaded assignment operator, the calling object must be created **before** calling this function).

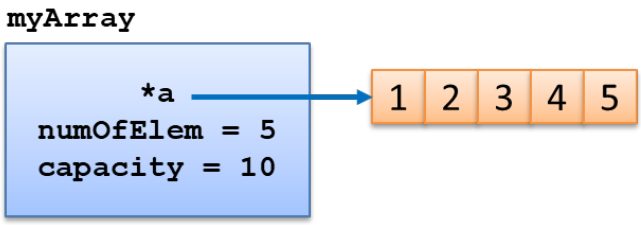
The **move constructor** passes an object of the same class, just like the **copy constructor**. Although you will not use the **const** modifier for the **parameter object** of the move constructor, because you are re-setting the parameter object, the functions will look the same to the compiler (the const modifier for parameters does not make any difference to the compiler). **Then how do we make these functions different**? By adding a **second reference** to the **parameter object** of the **move constructor (&&)**—note that the **syntax** for the implementation is the same as if it were a single reference. The **move assignment operator** will need a **double reference** as well.
The **call statement** to these functions needs to use the **move** function from the STL header **<utility>** (the **move** function from the **STL <algorithm>** header overloads this function). Note that the move function does **NOT** move anything, but it acts as a helper function. **Why?**
You may find some help by browsing the link below, but you will still need to think carefully about your implementation.
https://msdn.microsoft.com/en-us/library/dd293665.aspx

(Go to next page for a visual example.)

DArray Class Example

```
// Create a new object
DArray myArray(10);

// Insert elements…
myArray.insert(1);
myArray.insert(2);
…
myArray.insert(5);
```




```
// Use the move constructor to create another object.
// Note that you will need to use the "move" function
//    from the STL utility; also note that the "move"
//    function does not move anything, but it acts as
//    helper function. Why?
DArray yourArray = move(myArray);
```