In the first version of the class, a generic version of an ordered list is trying to be achieved. This ordered list should not have any empty spaces between numbers. Any empty spaces in the list will be located near the end of the list. I think the strengths of this class are that the empty spaces are in known positions, so it is easy to find the empty spaces while moving items in the array. A weakness is that this class must move nearly every item in the list, if an item is inserted in the first position. I think it will perform well if all the numbers added or removed are stored towards the end of the array. It will be more inefficient if the numbers are smaller and closer to the beginning.

In the second version of the class, a generic version of an ordered list is trying to be achieved. This ordered list should not have any empty spaces between numbers. Any empty spaces in the list will be located near the end of the list. The difference between this ordered list and the first is that the AddItem function for this list starts looking for the item from the end of the list. A strength of this is once again, the knowledge of where the empty spaces in the array and inserting numbers which should be near the end of the list would be easier to find. This method is once again inefficient for the inserting and removing items near the beginning of the list.

In the third version of the class, an ordered list which allows spaces between the items in the list is created. It is expected that the functions for add and remove methods will insert items in the list, leaving in the spaces. One strength is that there will be very few items moved in the list. A weakness is that this class has more comparisons in the function for add and remove.