Problem 1

Selection sort works by removing the smallest (or largest if sorting by largest) item from the list, and putting it in the next position in a new, now sorted list.

Sorted list										Original list							
										79	12	43	10	61	62	4	76
4										79	12	43	10	61	62	76	
4	10									79	12	43	61	62	76		
4	10	12								79	43	61	62	76			
4	10	12	43							79	61	62	76				
4	10	12	43	61						79	62	76					
4	10	12	43	61	62					79	76						
4	10	12	43	61	62	76				79							
4	10	12	43	61	62	76	79										

The above image shows a trace of the two arrays at every step in the Selection sort. Between each step in the image, the following must occur:

1. Copy the first item in the list into a variable "Smallest"
2. Compare that item to the item next to it.
   a. If the item next to it is smaller, Copy this new item into "Smallest"
3. If there is another item one further away from the item last compared to, compare to that item
   a. If that item is smaller, Copy this new item into "Smallest"
4. Repeat step 3 until there are no more items,
5. Remove the item it "Smallest" from the list it is in
6. Add the item in "Smallest" to the new, sorted array

Problem 2

Instead of copying data into a new list, the list could simply be reordered. By looking for the largest value (after the count of numbers already processed), and making it the head of the list, the list would end up sorted in an ascending order.
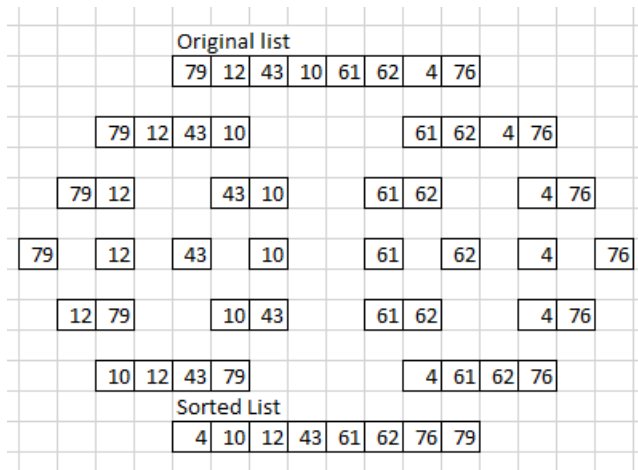
Problem 3

Merge sort works by first breaking down the list into individual elements, and then merging them into progressively bigger sorted lists.

Problem4

```cpp
ostream &operator<<(ostream &strm, const Clock &time)
{
    int printHour;
    string amPM;
    amPM = "AM";
    if(hour < 12)
    {
        printhour = hour - 12;
        amPM = "PM"
    }

    strm << setfill('0') << setw(2) << printHour << ":";
    strm << setfill('0') << setw(2) << getMin() << ":";
    strm << setfill('0') << setw(2) << getSec() << amPM << sendl;
    return strm;
}
};
```