

PART A:

```
public int compareCustomer(Customer other)
{
    int nameCompare = getName().compareTo(other.getName());
    if (nameCompare != 0)
    {
        return nameCompare;
    }
    else
    {
        return getID() - other.getID();
    }
}
```

PART B:

```
public static void prefixMerge(Customer[] list1, Customer[] list2, Customer[] result)
{
    int front1 = 0;
    int front2 = 0;

    for (int i = 0; i < result.length; i++)
    {
        int comparison = list1[front1].compareCustomer(list2[front2]);
        if (comparison < 0)
        {
            result[i] = list1[front1];
            front1++;
        }
        else if (comparison > 0)
        {
            result[i] = list2[front2];
            front2++;
        }
        else
        {
            result[i] = list1[front1];
            front1++;
            front2++;
        }
    }
}
```

Part A:	<code>compareCustomer</code>	3 points
----------------	------------------------------	-----------------

- +1 1/2 perform comparison
 - +1/2 attempt (must call `OBJ1.compareTo(OBJ2)`)
 - +1/2 correctly access and compare names
 - +1/2 correctly access and compare IDs
- +1/2 return 0 if and only if `this = other`
- +1/2 return positive if and only if `this > other`
- +1/2 return negative if and only if `this < other`

Part B:	<code>prefixMerge</code>	6 points
----------------	--------------------------	-----------------

- +1/2 initialize unique variables to index fronts of arrays
- +1 1/2 loop over arrays to fill result
 - +1/2 attempt (must reference `list1` and `list2` inside loop)
 - +1 correct (lose this if add too few or too many `Customer` elements)
- +1 1/2 compare array fronts (in context of loop)
 - +1/2 attempt (must call `compareCustomer` on array elements)
 - +1 correctly compare front `Customer` elements
- +1 1/2 duplicate entries
 - +1/2 check if duplicate entries found
 - +1/2 if duplicates, copy only one to `result` (without use of additional structure)
 - +1/2 update indices into both arrays (`list1` and `list2`)
- +1 nonduplicate entries
 - +1/2 copy only smallest entry to `result` (without use of additional structure)
 - +1/2 update index into that array only (`list1` or `list2`)

Note: Solution may use constants as returned from part A.

Usage: -1/2 `compareTo` instead of `compareCustomer` for `Customer` objects