List Position Class

Design a new type that uses a dynamic structure, which grows and shrinks as needed. The abstraction for this type is a pair of strings. For example: s c n g, d a r. This means that s c n and g have already been visited and the current position is at d.
In the following descriptions, consider each applied to this list position: s c n g, d a r

The List Position should allow users to perform the following functions:

1. Advance: Move one place to the right. In the above example the result of advance is s c n g d, a r.
2. Insert: Put in a new item at the current position. For example to insert w, the result is s c n g, w d a r.
3. Remove: Remove the item at the current position: s c n g, a r. Note: It is useful to return the data from the node you remove so the client can use that data. If the original list needs to be preserved, the client can insert the data back into the list.
4. Length_of_Remainder: 3.
5. Reset: Move back to the front: , s c n g d a r.
6. Swap_Remainders: Given 2 list positions, a b c, d e f and p q, r s t u, the result gives 2 new list postions: a b c, r s t u and p q, d e f.

Write the .h file, the .cpp file for this type.

To test the type, use it to do an insert sort. Perform the sort on 2 lists, then merge them, creating a final list in order. The client can choose whether the client wants to preserve the original lists or not.