(a) The sorted merge function uses chain iterators to move through the input lists `A` and `B`. At any time in the first `while` loop below, we are positioned at the first unused element in each of the lists `A` and `B`. The smaller of these is appended to the output list. If the appended element came from `A`, we move to the next element of *A*. Otherwise, we move to the next element of `B`.

```
template<class T>
void Merge(const Chain<T>& A,
           const Chain<T>& B, Chain<T>& C)
{// Merge from A and B to get C.
   ChainIterator<T> a,  // iterator for A
                    b;  // iterator for B
   T *DataA = a.Initialize(A);
                    // pointer to an element of A
   T *DataB = b.Initialize(B);
                    // pointer to an element of B
   C.Erase(); // empty out chain C

   // merge until one of A and B is empty
   while (DataA && DataB) {
      if (*DataA <= *DataB) {// A goes next
         C.Append(*DataA);
         DataA = a.Next();}
      else {// B is smaller
         C.Append(*DataB);
         DataB = b.Next();}
      }

   // append the rest
   // at most one of A and B is nonempty now
   if (DataA) while(DataA) {// A is not empty
                 C.Append(*DataA);
                 DataA = a.Next();
                 }
   else while(DataB) {// B is not empty
           C.Append(*DataB);
           DataB = b.Next();
           }
}
```

(b) We shall do the analysis under the assumption that the merge is successful (i.e., no exception is thrown). In each iteration of the first `while` loop, we move one node right either in `A` or `B`. So, the complexity of this loop is O(length of `A` + length of `B`). The complexity of the second `while` loop is O(length of `B`) and that of the third loop is O(length of `A`). The call to `Erase` takes Θ(length of initial `C`) time. Also, Ω(length of `A` + length of `B`) time is spent constructing the final `C`. So, the overall complexity is Θ(sum of initial lengths of the three lists `A`, `B`, and `C`).

(c) The codes and output are in the files `cmerge1.cpp` and `cmerge1.out`.