(a) A chain may be reversed by reversing the direction of the pointers in each node. For this, we use three pointers to march through the chain left to right. `current` points to the node whose pointer (link) we are about to reverse; `last` points to the node on its left (`last` is zero in case `current` is the first node); and `next` points to the node at the right of `current` (`next` is zero in case `current` is the last node of the chain). The link in `current` is changed from `next` to `last`. Then `last`, `current`, and `next` are advanced one node to the right. The code for the member function to reverse a chain is

```
template<class T>
Chain<T>& Chain<T>::Reverse()
{// In-pace reversal of the chain.
   ChainNode<T> *last = 0,  // last node
                *current = first,
                           // current node
                *next;     // next node
   while (current) {
      // change pointer direction
      next = current->link;
      current->link = last;

      // move to next node
      last = current;
      current = next;
      }

   first = last; // new first node
   return *this;
}
```

(b) The complexity is Θ(`length`) as the `while` loop iterates `length` times and each iteration takes $\Theta(1)$ time.

(c) The codes are in the files `achain.h` and `reverse4.cpp`. The output is in `reverse4.out`.