Example - Reverse a Singly Linked List, Version 1

Make a new list, use basic operations: Copy $L$ to a second list $R$ one item at a time. Because items are removed from the front of $L$ and placed at the front of $R$, we are essentially using $R$ as a stack where inserting at the front is pushing onto $R$. 

$$R := \text{CREATE}$$

$$\text{while (not EMPTYS(L)) do begin}$$

$$d = \text{data stored in the first vertex of } L$$

Delete the first vertex of $L$.

Insert a new vertex at the front of $R$ with data $d$.

$$\text{end}$$

$$\text{DESTROY(L)}$$

$L := R$

**Complexity:** $O(n)$ time and $O(1)$ space in addition to the space consumed by the list vertices.
Example - Reverse a Singly Linked List, Version 2

Use “low-level” operations to reverse in-place: Reverse each next pointer and finish by exchanging the first and last pointers of the header.

\[
\begin{align*}
\text{previous} & := \text{nil} \\
\text{current} & := L->\text{first} \\
\text{while } (\text{current} \neq \text{nil}) \text{ do begin} \\
\quad & \text{temp} := \text{current}->\text{next} \\
\quad & \text{current}->\text{next} := \text{previous} \\
\quad & \text{previous} := \text{current} \\
\quad & \text{current} := \text{temp} \\
\text{end} \\
L->\text{last} & := L->\text{first} \\
L->\text{first} & := \text{previous}
\end{align*}
\]

**Complexity:** \(O(n)\) time and \(O(1)\) space in addition to the space consumed by the list vertices.
Example - Reverse a Singly Linked List, Version 3

Again, use “low-level” operations to reverse in place: Traverse the list and change each pointer to point to the current first item (i.e., combine the idea of version 1 with the low level operations ov version 2).

\[
\begin{align*}
  v & := L->first \\
  L->first & := \text{nil} \\
  \text{while (v \neq \text{nil}) do begin} \\
  & \quad \text{temp} := v->next \\
  & \quad v->next := L->first \\
  & \quad L->first := v \\
  & \quad v := \text{temp} \\
  \text{end}
\end{align*}
\]

**Complexity:** $O(n)$ time and $O(1)$ space in addition to the space consumed by the list vertices.