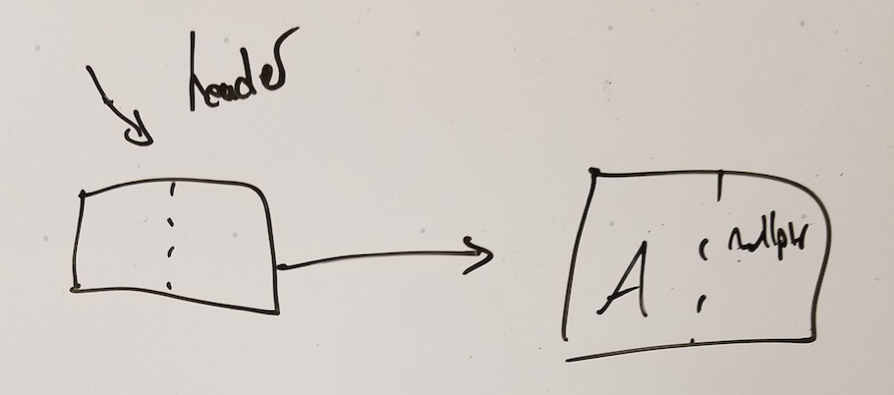
Jake Goldstein

HW 06 Written

1. Picture:



1. For each of the following, determine if the code compiles
   1. Should not compile because normally when we write code like this we are doing in terms of adding ints to an iterator but here it’s a little all over the place and therefore unclear what is actually meant so it wont compile
   2. Should not compile because comparing the list iterators with the ‘<’ operator I wrong.
2. For each of the following, determine if the iterator is valid.
   1. This is valid because vItr is pointing to something before the element that go erased.
   2. This is invalid because the item that was erased if before where vItr is pointing to.
   3. This is vald because lItr is an iterator for a list which generally only becomes invalidated if the element that that iterator is pointing to is erased
   4. This is vald because lItr is an iterator for a list which generally only becomes invalidated if the element that that iterator is pointing to is erased
3. Which of the following code snippets are valid? If the code snippet is invalid, state why.
   1. This is vald because lIter is an iterator for a list which generally only becomes invalidated if the element that that iterator is pointing to is erased. So adding more elements to the list is validated.
   2. This is invalid because we are talking about a list here and we do not have random access. Meaning that we are not able to subtract the two iterators from each other.
   3. Valid
4. Two problems here. The first is that the first element in the list never gets checked because you are starting at prev = header→next, and the first element that gets checked is prev→next→data. Secondly, you never test to see that you are allowed to look at what is at prev→next→data. Meaning that prev may not be a nullptr, but prev→next may be pointing to a nullptr, but then you check what is there with prev→next→data.