Jake Goldstein

HW 03 Written

1. The first cout ≪ is going to print 6 because c.end( ) - c.begin( ) will give the size of the vector which is 13. Then when that is put over 2, it becomes 6. So when we add c.begin() to 6, we are now at the 7th element, which is 6.

The second cout ≪ is going to print a 2, because the erase did not effect itr1 because it was before the erased element, which was c.being()+2.

The next cout ≪ is going to print out 5, because the element at itr2 was erased, which was 4, so now the 5 is sitting at that same index.

The second to last cout ≪ is going to print 9 because all of the elements of the vector were moved over once when the 5th element was erased.

The last cout ≪ is going to print 7 because now the size is 12, so 12/2 is still 6, so when we move over from the beginning 6 times we are at the 7th index which is now 7.

So the output is 6, 2, 5, 9, 7

1. O(n)
2. Given the following code snippet… Assign values to the iterators….
   1. itrStart = a.begin();

int mid = (a.end() - a.begin())/2;

itrMid = a.begin() + mid;

* 1. int mid = (a.end() - a.begin())/2;

itrMid = a.begin() + mid;

itrEnd = a.end();

1. For each code snippet below state either why the code won’t compile/run, or state what is printed by the code snippet.
   1. 4
   2. This will not compile because it’s a list and you can’t jump spots on the list by adding 3.
   3. 5
   4. This will not compile because the iterator that we are trying to make here is for chars and the vector is for ints.