Jake Goldstein

HW 04 Written

1. O(n)
2. O(n)
3. Any iterator that is before the index which was erased will sill be valid, and anything after that index (inclusive will be invalid). Valid: (start, erased) Invalid: (erased, end-1). (there is a minus 1 there because now the size of the vector is one smaller than it was beforehand)
4. 0. When the iterator is created it is pointing to some location on the heap, however when there is no more spare capacity in the vector it is created in a new space on the heap, so the iterator is no longer pointing to the ‘correct’ array, so it puts out a garbage value of 0.
5. Since we initialized SPARE\_CAPACITY to be 2, this will output 112. Essentially what is happening is that when we make the vector with a parameter of 3 (or a capacity of 5) it works as we wanted it to, as if with the explicit. However when the v = 110 line is called, without the keyword explicit there, the compiler thinks that we want to make the vector of size 110 (or capacity 112).
6. The C++ STL has many functions and functors. Here is your chance to try some of them. In a program when you use an STL algorithm add #include, and when you use an STL functor add #include.
   1. Copy(a.begin(), a.begin()+6, d.begin());
   2. Count(B.being(), b.end(), 1);
   3. count\_if(b.begin( ), b.end, bind1st(not\_equal\_to( ), 1));
   4. vecItr = find(a.begin(), a.end(), 5);

cout ≪ \*vecItr;

* 1. vecItr = find\_if(c.begin(), c.end(), bind2nd(greater(), 2));

cout ≪ \*vecItr;

* 1. reverse(c.begin(), c.end());
  2. sort(b.begin(), b.begin()+4);

sort(a.begin, a.end(), greater<int>());

* 1. random\_shuffle(a.begin(), a.end());