Key Modification in Associative Containers Solutions

Key Modification

- Write a C++14 program that changes the key of an element in an associative container
- Are there any drawbacks to your solution?
  - The keys in an associative container are const
  - The best solution is to erase the element and insert a new element with the new key and the original value
  - This is inefficient as it involves several copy/move operations
  - The modified element must be reallocated in memory
  - The erase and insert operations may cause the tree to rebalance

Key Modification in C++17

- Write the same program again, using C++17 features to make it as efficient as possible
- What are the advantages of this approach?
  - Does not call copy/move constructor of the value
  - Does not allocate or deallocate any memory
  - insert() will make one call to node_handle's move constructor
  - The insert() call may cause a tree rebalancing

Transfer of Element Ownership

- Is it possible to transfer an element from one associative container to another?
  - Yes
  - The containers must be the same type (or multi-key to unique-key and vice versa)
  - The key and value types must be the same (and the allocator types, if not using the default)
- Write a program to illustrate your answer