Removing Algorithms Continued Solutions

remove_if()

- Describe the remove_if algorithm function
  - remove_if() allows us to provide our own predicate
  - This predicate will be used instead of operator == to decide which elements to remove
- What arguments does remove_if() take?
  - remove_if() takes an iterator range and a predicate function
- Write a simple program which uses remove_if()

remove_copy()

- Describe the remove_copy algorithm function
  - remove_copy() will perform a copy operation, omitting the matching elements
- What arguments does remove_copy() take?
  - remove_copy() takes an iterator range, the value to be removed, and the destination
- Write a simple program which uses remove_copy()

remove_copy_if()

- Describe the remove_copy_if algorithm function
  - remove_copy_if() allows us to provide our own predicate
  - This predicate will be used instead of operator == to decide which elements to remove
- What arguments does remove_copy_if() take?
  - remove_copy_if() takes an iterator range, a predicate function, and the destination
- Write a simple program which uses remove_copy_if()

unique()

- Describe the unique() algorithm function
  - unique() removes duplicate adjacent elements
  - unique() behaves similarly to remove() in that it does not destroy any elements
  - Optionally, we can provide our own predicate
  - This predicate will be used instead of operator == to decide which elements to remove
- What arguments does unique() take?
  - unique() takes an iterator range and an optional predicate function
- Write a simple program which uses unique()

unique() with Predicate

- Modify your solution so that the unique() call takes a predicate argument

unique_copy()

- Describe the unique_copy algorithm function
  - unique_copy() will perform a copy operation, omitting duplicated elements
- What arguments does unique_copy() take?
  - unique_copy() takes an iterator range and the destination
- Write a simple program which uses unique_copy()