# CS 304
# Program 2: Container Classes

## Due Friday, Sept. 27, 10:00 pm
## (but see due date for “Show me that you got this far” on page 2)

Note that I do not anticipate being available Friday, Sept 27, later than my regular office hours (2:40-3:30). I suggest that you not wait until the last minute to complete this project.

This project is worth a total of 80 points, or 8% of your course grade

### Goals

- Implement and test a container class in C++ that uses dynamic memory.
- Document class value semantics, and class invariants.
- Demonstrate the use of typedefs for value_type and size_type to reduce platform dependence.
- Demonstrate the use of exceptions to signal errors.

### Reading

Main & Savitch:
- Ch. 3.1
- Ch, 4.1-4.4
- Page 204 (overloading [])
- Pages 227-230 (**const** keyword with a pointer)
- Appendix L (but note that there are syntax errors in the examples)

## Overview

This programming project is based on the Bag class described in your textbook in chapters 3 and 4 and on programming project 7 suggested on page 150 (bag with receipts).

Implement the **bag_with_receipts** class as suggested on page 150, except
- Place the class in the namespace **cs304_bag**.
- Remove the **erase** and **erase_one** functions.
- Remove the **operator +** and **operator** += functions.
- Modify the **reserve** function to ensure that it does not change the capacity to a size smaller than the largest receipt currently in use.
- Add a **remove** function. **remove** should take a receipt as a parameter and return a copy of the Item stored for that receipt value, if present, or return null. In either case, that receipt location is no longer in use.
- Override the [] operator (array index) to return a copy of the Item at the given receipt number, if present. If not present, throw an **out_of_range** exception.This differs from **remove** in that operator [] does not remove the Item (and **remove** does not throw an exception).
- Override the [] a second time as a **const** member function that returns a const pointer to the Item. It should otherwise behave the same as the non-const version.

Be sure to update the Value Semantics, Dynamic Memory Usage, and invariant statements in light of your changes.

## Details

The **bag2** header file (from Chapter 4 of your textbook) is provided as a starter file. Complete and test the implementation of the class.

### Header file (10 points total)

Modify the header file to make the changes identified above. (5 points)

Update the API, including the value semantics and dynamic memory usage statements. (5 points)

### Basic member functions (17 points total)

Modify the implementation file to
- Remove all functions no longer needed (2 points)
- Update the invariant statement (5 points)
- Update the constructor to support the new functionality (5 points)
- Update the destructor, and implement a test to demonstrate that it works (5 points)

## Show me that you got this far (3 points total)

Bring your running, tested program to my office (or ask me to come to your computer in the lab) and show me that you have gotten this far no later than **Friday, Sept. 20, 3:30 pm**.
Note: you must talk to me in person; e-mailed submissions will not be accepted unless prior arrangements have been made.

### Remaining member functions (35 points total)

Implement/modify and test the remaining functions
- insert (3 points)
- size (2 points)
- reserve (5 points)
- remove (5 points)
- non-const [] (10 points)
- const [] (10 points)

### Quality (15 points total)

- Code conforms to (updated-for-assignment 2) C++ coding standards (15 points)

## Submitting

Zip up your VS project and submit it through Moodle.