**The City College of NY**

**Computer Science Grove School of Engineering**

**Computer Science Department**

**Data Structures & C++**

**CSC 21200(LM-24126)**

Instructor: Yedidiah Solowiejczyk

Dates: **Jan 26 – May 23, 2023**

**Time: Tu & Th 10:00AM – 11:40 AM**

Location: **Hybrid**(**Zoom Remote**) & TBD

Sessions: 30

-------------------------------------------------------------------------------------------------------------------

**COURSE Presentation Mode:**

This course is being offered **Hybrid** **Synchronous** this **Spring** semester.

**COURSE DESCRIPTION:**

This course will provide the student with a thorough knowledge and programming training in a variety of different Data Structures (Vectors, Matrices, Linked Lists, Binary Trees, etc). The course will also cover a thorough review of Templates and the Standard Template Library (STL) which provides a wide variety of data containers, iterators, and library of algorithms.

The course provides a wide variety of common programming problems and the use of STL's ability to provide generic, efficient solutions to these problems is emphasized.

**COURSE OBJECTIVES**

At the completion of the course, the student will be able to:

* Use collection of containers to implement real world data storage problems.
* Understand the use of a variety of different STL containers, iterators, algorithms to quickly develop efficient solutions to data storage problems.

**COURSE DISCLAIMER:**

Students who participate in this class with their camera on or use a profile image are agreeing to have their video or image recorded solely for the purpose of creating a record for students enrolled in the class to refer to, including those enrolled students who are unable to attend live.  If you are unwilling to consent to have your profile or video image recorded, be sure to keep your camera off and do not use a profile image. Likewise, students who un-mute during class and participate orally are agreeing to have their voices recorded.  If you are not willing to consent to have your voice recorded during class, you will need to keep your mute button activated and communicate exclusively using the "chat" feature, which allows students to type questions and comments live.

**COURSE PREREQUISITES:**

* Prerequisite: [CSc 10300](file:///Users/yedidiahsolowiejczyk/Desktop/CCNY/10300.html),  [CSc 10400](file:///Users/yedidiahsolowiejczyk/Desktop/CCNY/10400.html)

**CALENDAR:**

* **Thu, Jan 26, 2023 – Classes begin**
* **Tu, Feb 21, 2023 - Classes follow Monday schedule**
* **Thur, Apr 6 - Thur, Apr 13 – No Classes (Spring Recess)**
* **Frid, May 12 - Last day to file for P/NC option**
* **Tu, May 16 – Last Day of Classes**
  + **Course Withdrawal period ends (Last day to withdraw from a course with a grade of "W");**
* **Thur, May 17 - Tu, May 26 – Final Exams**
* **Tu, May 23 - End of Spring Term**
* **Fri, May 26 – Final Grade Submission**

**TEXTBOOK:**

**Required Textbook & Readings:**

* **There is no required textbook purchase for the course.**
* **However, the following texts are provided to you electronically as supplementary readings:**
  + **“*Thinking in C++”* (Vol 1 & Vol 2) - Bruce Eckel**
  + **“*Data Structures Using C and C++”* – Y. Langsam, M. Angestein, A. Tanenbaum**
  + **“*Data Structures in C++ Using the Standard Template Library*” – Timothy Budd**
* **You will also be provided with a series of articles posted on the web that describe the topics that we will be covering this semester.**

**RECOMMENDED Reading Notes (Not Mandatory):**

* **GeekForGeeks Notes**:

#0 – Data Structures & Algorithms

#1\_Introduction to Data Structures.docx

#2\_String class in C++.docx

#3\_Vector in C++.docx

#4\_Linked List C++.docx

#4\_bis\_Linked List \_C++.docx

#5\_Doubly Linked List C++.docx

#6\_Binary Tree C++.docx

#7\_Hashing.docx

#8\_List in C++ Standard Template Library (STL).docx

#9\_STL\_Algorithm Library.docx

**Container Notes**

* What are Containers in STL
* Vector Container
* LIST Container
* Deque Container
* Queue Container
* Stack Container
* Introduction to Iterators
* Map Container
* Bitset Container
* Overview of Algorithms in STL
* Summary of Algorithms in STL
* Functors in STL.docx
* Graph implementation using STL

**ASSIGNMENTS:**

* There will be reading assignments each week covering the material discussed in class.
* There will be approximately***10 programming homework assignments*** consisting of **two** (2) programming exercises which will demonstrate the student’s knowledge and use of Data Structures in C++.
* Students will submit the HW assignment for each week as **HWxPy.cpp** file and screenshot of the results. (x – week # and y is 1 or 2; e.g. ***HW1P2.cpp*** )
* The **HW assignments** will be due **two (2) weeks** after the assignment date. Late HWs will be **penalized** for lateness.
* ***Two (2) Take-Home*** exams, consisting of (2) questions, will be handed out on:
* **Week 5** which must be completed by end of **Week 7.**
* **Week 10** which must be completed by end of **Week 12.**
* A **Take-home Final Exam**, consisting of **two (2)** questions, will be handed out on **Week 13** which will be due on **Week16**.
* The **On-Line** Final exam will be ***two hours*** long and consist of programming **two (2)** questions and compiling them which will demonstrate the knowledge of Data Structures. The students will submit separate **\*.cpp** files + **screen shots** of output data generated by these programs. The On-Line Final Exam will take place somewhere between **May 17 – May 23(TBD)**

**GRADING:**

The Final Course Grade will be based as follows:

* **Homework Assignments - 30%**
* Take-home Exam#1 - 15%
* Take-home Exam#2 - 15%
* Take-home/Final - 15%
* **On-Line Final Exam - 20%**
* **Class Attendance/Participate - 5%**

|  |  |
| --- | --- |
| **Grade** | **Point cutoff** |
| A | 90 - 100 points |
| B+ | 85 – 89 points |
| B | 80 – 84 points |
| C+ | 75 - 79 points |
| C | 70 – 74 points |
| D | 60 – 69 points |
|  |  |
| F | **Lower than 60 points** |

**email address:**

**ysolowiejczyk@ccny.cuny.edu**

[**ysolowie@yahoo.com**](mailto:ysolowie@yahoo.com)

**CCNY BLACKBOARD**

* + Class Notes, emails, Syllabus and other programming material

**COMPILERS:**

* Microsoft’s Visual C++ 2010 - <http://www.microsoft.com/express/vc/>
* Apple C++ - <http://developer.apple.com/tools/xcode/index.html>
* Borland C++ - <http://www.codegear.com/downloads/free/cppbuilder>
* Bloodshed Development – C++ - http://www.bloodshed.net/devcpp.html
* <http://www.research.att.com/~bs/compilers.html>
* <http://www.thefreecountry.com/compilers/cpp.shtml>

**ATTENDANCE POLICY**

Student attendance records will affect their final grade. Students should inform the instructor of any anticipated absences or last minute absences. Homework assignments and take-home and final exams should be handed on a punctual basis.

**ACADEMIC INTEGRITY:**

*Academic* ***dishonesty*** *is* ***prohibited and not tolerated*** *in The City University of New York. Penalties for academic dishonesty are clear and will be immediately enforced. Academic acts of dishonesty such as plagiarizing someone else’s code or exams will* ***trigger an immediate F (zero-score) for the first offense followed by an F in the course in the second offense****. The school and department will pursue suspension or expulsion for serious offenders. The school as well as myself have the tools to discover dishonest behavior which will not be tolerated. We encourage all students to work by themselves in producing their homework outputs as well as exams.*

*For more details please visit* [*https://www.cuny.edu/about/administration/offices/legal-affairs/policies-procedures/academic-integrity-policy/*](https://www.cuny.edu/about/administration/offices/legal-affairs/policies-procedures/academic-integrity-policy/)

***Table of Contents***

1. Week: Review of C++ - (Th, Jan 26) 8

2. Week: Continue Review of C++ - (Tu, Jan 31 – Th, Feb 2) 9

3. Week: Review Inheritance, Composition - (Tu, Feb 7 – Th, Feb 9) 10

4. Week: String Class (Tu, Feb 14 – Th, Feb 16) 11

5. Week: Linked Lists (Tu, Feb 21 (*no classes*) - Th, Feb 23) 12

6. Week: Trees - (Tu, Feb 28 - Th, Mar 2) 13

7. Week: Hash Tables - (Tu, Mar 7 – Th, Mar 9) 14

8. Week: Templates - (Tu, Mar 14 – Thu, Mar 16) 15

9. Week: The Standard Template Library - (Tu, Mar 21 – Th, Mar 23) 16

10. Week: The Standard Template Library - (Tu, Mar 28 – Th, Mar 30) 18

11. Week: STL Containers - (Tu, Apr 4 – Th, Apr 5(*no classes*) ) 20

12. Week: Spring Recess – (Tu, Apr 11 – Th, Apr 13) 21

13. Week: STL Iterators - (Tu, Apr 18 – Th, Apr 20) 22

14. Week - Special Containers - (Tu, Apr 25 – Th, Apr 27) 23

15. Week - STL Algorithms - (Tu, May 2 - Th, May 4) 24

16. Week - STL Algorithms - (Tu, May 9 - Th, May 11) 26

17. Week – Review Course Material (Tu, May 16) 28

# Week: Review of C++ - (Th, Jan 26)

* **Review**
* **Syllabus**, Reading Assignments, HW Assignments,
* Midterm (Take-Home) Exam, Final Exam,
* Grading, Class Attendance,
* Academic Integrity, Office Hours
* **Overview of the C++ language & Need for Data Structures**
* **Variables, Types, Operators, Keywords**
  + - * Declaring variables in C++
      * Operators & Precedence (+,-,\*,/,++,--,etc.)
      * Keywords C/C++, Punctuation
      * **Quick Overview of Containers in C++**
* **Arrays & Strings**
* **Need for Other Containers**
  + - * **Functions**
* Function Basics
* Passing/Returning Parameters of Functions
* Function Overloading

**Readings:**

**“Data Structures Using C and C++” - Langsam, Angestein, Tanenbaum**

Chapter 1: pp 1-8

**GeeksforGeeks: #0 “Data Structures & Algorithms”**

**HW#1:**

1. **What is the decimal-base value of the following octal/hex constants?**
   * **040**
   * **0X40**
   * **0XFF**
   * **0XCAB**
2. **Write a C++ program that calculates sum of the first two values above using the twos-complement algorithm**
   * **int Prod(int X, int Y)**

**int Result = Prod( (int)042, (int)0X42);**

**return Result;**

# Week: Continue Review of C++ - (Tu, Jan 31 – Th, Feb 2)

* **Pointers and Memory Management**
* Declaring Pointer Variables
* Memory Management & Dynamic Memory Allocation
* **Review of Data Structures** 
  + Declaring data structures & Initializing data structures
  + Accessing data structure member data via **“.”** operator
* **Interface functions** 
  + - Pass By value
    - Pass By pointer
* ***Introduction to Objects***
* What is an object
* ADT(Abstract Data Types)
* Object Design
* ***Class Properties***
* Declaring a Class
* Instantiation of Class
* Constructors/Destructors/Copy Constructors
  + Protecting data
* Inline functions
* Function Overloading
* Operator Overloading( +,-,\*,/, ++, =, ==, <<, >>, etc.)
* **Example: Vector Class**

**Readings:**

***Eckel(Vol 1) Chapter 13 – Dynamic Object Creation (pp 580-586)***

**GeeksforGeeks: “#1\_Introduction to Data Structures”**

**HW#2:**

1. Write a program having a **String class** that contains a function that can reverse the order of the characters in a string. A ***palindrome*** is a word or group of words that read the same forward and backward. For example, “madam” or “wow”. Write a program that takes a string argument from the command line and returns TRUE if the string is a palindrome.
2. Write a function that takes as argument an array of integers and the size of the array and returns the average of the input values.
   * **Int = avrg(int \* array, int size)**

# Week: Review Inheritance, Composition - (Tu, Feb 7 – Th, Feb 9)

* **Inheritance & Composition**
* What is inheritance
  + Single vs. Multiple
* Inheritance Syntax
* What is composition
* Instantiating Derived & Composite Objects
* **Example: Class Matrix (Vector of Vectors)**

### Readings:

***Eckel(Vol 1) Chapter 14 - Inheritance & Composition, (pp 613 – 625)***

**HW#3:**

1. Write a program containing a base ***Animal*** class that can contains member functions that can output (i) ***”I can eat”***, (ii) ***“I can sleep”*** as well as a **showData()** member function that outputs the ***weight*** and ***age***. A ***derived*** class ***Dog*** contains a function that displays ***“I am a dog”*** *and can output the dog species’ name (e.g. “bulldog”). Both the base class and derived class must have appropriate constructors, destructors and assignment=() operators.*
2. A Date class is described by **month(m), day(d), year(y).** An Employee is described by a unique data structure having a **name** (e.g. “George”), **salary**(75000.25) and **date\_of\_birth** (06, 10, 1998). Calculate the number of days between his day of birth and (09, 20, 2022). Assume that the Employee object contains a Date sub-object)

# Week: String Class (Tu, Feb 14 – Th, Feb 16)

# 

* **Strings**

1. String Types & Characteristics(C-Type vs. C++ Strings)
2. Constructor, Destructor & Copy Constructor
3. Size & Capacity
4. Element Access [ ]
5. Comparing Strings
6. Copying Strings
7. String Concatenation (+)
8. Input/Output Operators
9. Searching and Finding
10. Inserting Characters in a String
11. Numeric Conversions (atoi, atof, etc)
12. Performance

### Readings:

#### GeeksForGeeks: #2\_String Class in C++

**HW#4:**

1. Write a short program that reads phrases from the input stream and stores each new line into an array of C++ strings. For each new line, the program computes the numeric sum value of each phrase terminated with ‘\n’. For the following section of the Gettysburg Address store the numeric values of each phrase into a int cnt[32] array.

“Four score and seven years ago our fathers brought forth upon this continent, a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal.

Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure.

We are met on a great battle-field of that war. We have come to dedicate a portion of that field, as a final resting place for those who here gave their lives that that nation might live.

It is altogether fitting and proper that we should do this...”

1. Two words are said to be anagrams if they contained the same characters but in different order – for instance “rescue” and “secure”. Write a program that takes to strings and determines if they are anagrams of each other (hint – sort both words and check if they match)

# Week: Linked Lists (Tu, Feb 21 (*no classes*) - Th, Feb 23)

* **Linked Data Structures**
* **Link List Nodes**
  + Creating Nodes
* **Linked Lists**

Inserting a Node at the Head of a List

Inserting and Removing Nodes Inside a List

Searching a Linked List

Doubly Linked Lists

Inserting a Node at the Tail of a Doubly Linked List

Inserting & Deleting a Node from a Doubly Linked List

* **Derived Containers from Link Lists**
  + **Stacks**
  + **Queues**

### Readings:

#### GeeksForGeeks:

#### #4 “Linked List (Introduction)”

#### #4\_bis “Linked List | Set 2 (Inserting a node)”

#### #5 “Doubly Linked List (Introduction and Insertion)”

**HW#5:**

* 1. Write a program that creates a singly link list of used automobiles containing nodes that describe the model name (string), price(int) and owner’s name. The program should create a list containing 12 nodes created by the user. There are only three types of models (BMW, Cadillac, Toyota) and the prices range from $7500 – $20,500. The program should allow the user to provide
* **Print a printout of all cars contained in the list (model, price, owner) – Print(list\_ptr\*)**
* **Provide a histogram(global array) of all cars in the list portioned into $500 buckets**
* **Find all cars that cost less $7500 - find\_if( list\_ptr\*, int )**
* **Calculate the average price of the cars contained in the list – avrg(list\_ptr\* )**
* **Provide the details for all cars more expensive than the average price**
* **Remove all nodes having a price less than 25% of average price – remove(list\_ptr\*, int average)**
* **Print a printout of all cars contained in the updated list (model, price, owner)**
  1. **Write a program that creates a stack container from a Singly Linked List. Demonstrate using it by creating a stack of 9 car models.**

**Take-home Exam#1 will be handed out – return on Week 7**

# Week: Trees - (Tu, Feb 28 - Th, Mar 2)

* **Trees**
* **Binary Tree Nodes**
  + Creating Nodes (Left Child vs. Right Child)
* **Binary Trees**
  + Tree Transversal
    - Pre-Order
    - Post-Order
  + Binary Search Trees
  + Inserting a New Node
  + Removing a Node

### Readings:

#### GeeksForGeeks:

#### #7 “Binary Trees”

#### HW#6:

1.Write a program that implements a binary tree having nodes that contain the following items: (i) Fruit name (ii) price per lb. The program should allow the user to input any fruit name (duplicates allowed), price. The root node should be initialized to {“Lemon” , $3.00}. The program should be able to do the following tasks:

#### create a basket of 16 fruits/prices

#### list all the fruits created (name/price)

#### calculate the average price of the basket

#### print out all fruits having the first letter of their name >= ‘L’

1. Execute the program above for the case where the input fruits are: apple, banana, cherry, date, elderberry, fig, guava, huckleberry, Indian berry, jackfruit, kiwi, lemon, mango, nectarine. How many searches are necessary to find kiwi and nectarine. Does this search obey a log behavior?

# Week: Hash Tables - (Tu, Mar 7 – Th, Mar 9)

* **Big *O-Notation***
* **Hash Table Abstraction**
* **Hash Functions**
* **Collision Resolution Using Buckets**
  + **Asymptotic Analysis of Hash Table Operations**
* **Hash Table Sorting**
* **Hash Data Type**

### Readings:

“**Data Structures Using C and C++**”

* **Hashing - Section 7.4**

**GeeksForGeeks:**

* **#7\_Hashing Data Structure**

**HW#7:**

1. **You are given a list of days-of-year (1-365) that have a corresponding daily temperature reading (0 -100). Write a program that requests from user a unique day-of-year with a corresponding random temperature (0-100). The day-of-year value should be hashed into a hash table (size 17). Print al the entries of the hash table and corresponding daily temperatures**
2. **Print out all days-of-year entries that are a multiple of 3**

**Deadline for submitting Exam#1**

# Week: Templates - (Tu, Mar 14 – Thu, Mar 16)

* **Templates** 
  + Overview of functions in C++
  + Function Templates
    - Template Definitions
    - Overloading Function Templates
    - Defining a Class Template
  + Syntax for Class Templates
  + Non-inline function definitions
  + Header files
  + Constants in templates
  + Template Parameters
  + Template Type Parameters
  + Writing Generic Programs
* **Instantiation**
  + Function-Template Explicit Arguments
* **Template Compilation Models**
* **Class Template Members**
  + Class template member Functions
  + Template Arguments for Non-Type Parameters
  + Static Members of Class Templates
* **Template Specializations**
  + Specializing a Function Template
  + Specializing a Class Template
  + Specializing Member but not the Class
  + Class-Template Partial Specializations

**Readings:**

**Eckel Chapter 16(Vol 1) – pp 727-735**

**HW#8:**

1. **Create template for the array class so that one can create an array of arbitrary size for ints, floats, chars, etc. The class should contain default/parametrized constructor, copy constructor, destructor, overloaded assignment operator, index operator, insertion and extractor operators.**
2. **Create a template for a complex number object where the real/imaginary components can be int, float, double). The class should contain constructor, destructor, copy constructor, assignment operator, overloaded operator+( ), overloaded operator\*( ).**

# Week: The Standard Template Library - (Tu, Mar 21 – Th, Mar 23)

* **Introduction to STL (Standard Template Library)**
  + Origins of STL
  + Containers, Iterators & Algorithms
* **Sequence Containers Example**
  + **Vectors** 
    - Abilities of Vectors (Member Functions)
      * Constructor (default, parametrized)
      * Size vs. Capacity
      * Member Functions: push\_back & pop\_back, insert & erase
      * sort, swap, =, [ ], etc.
    - Advantages & Disadvantages of Vectors
      * Insertion/Deletion
      * Appending
    - Iterators and Vectors
    - Passing Vectors to Functions
    - Exception Handling
    - Vectors of User Defined Object
      * Examples: Rational Numbers, Strings, Spatial Coordinate Points

**Readings**:

* **Containers:**
  + **Introduction to STL.docx**
  + **What are Containers in STL.docx**
  + **Vectors.docx**
  + **Eckel (Vol. 2) – Ch. 15 Freely-available STL extensions**

**HW#9:**

#### Create program that creates arrays of size 8 for the following data types(use STLvector template):

* + - **ints, floats, chars, strings, pairs**
    - **initializes these arrays with user-defined values**
    - **sorts the arrays by ascending value**
    - **reverses the values of the sorted arrays**
    - **passes the arrays to a function Sum( ) that calculates the sum of all the edata elements**

#### Write a program that contains a “Dog” class whose data consists of

#### int Id

#### static int cnt that keeps track of how many dogs have been created.

#### The Dog class should contain default constructor that sets the Id value with the updated vale of cnt and prints a message: cout << “dog #Id:” << X[i].Id.

#### The Dog class should also have a copy constructor that updates cnt and assigns it to the Id# of constructed dog.

#### The copy constructor should output the Id# of the constructed object. The Dog class should have a destructor that reduces the cnt value each time a “Dog” goes out of scope.

#### The destructor should print the Id# of the “dog” that goes out of scope.

* + - Create a vector array of dogs(size 8).
    - Output the Id values of all the dogs
    - After creating the Dog array, insert a new dog into location #5.
    - Explain the stream of outputs that this problem creates.

# Week: The Standard Template Library - (Tu, Mar 28 – Th, Mar 30)

* **Sequence Containers(Cont.)**
  + **Lists** 
    - Abilities of Lists(Member Functions)
      * constructor
      * push\_back & pop\_back, insert & erase
      * sort, swap, =, etc.
    - Advantages of Lists
      * Insertion/Deletion
      * Appending
    - Lists of User Defined Object
      * Examples: Rational Numbers, Strings, Spatial Coordinate Points
* **Deques** 
  + Abilities of Deques(Member Functions)
    - * constructor
      * push\_back & pop\_back, insert & erase
      * sort, swap, =, etc.
  + Advantages & Disadvantages of deques
    - Insertion/Deletion
    - Appending
  + Examples of Using Deques
* **Bitsets**

constructors

[**operator[]**](https://cplusplus.com/reference/bitset/bitset/operator%5b%5d/)

[**count**](https://cplusplus.com/reference/bitset/bitset/count/) - Count bits set (public member function)

[**size**](https://cplusplus.com/reference/bitset/bitset/size/) - Return size (public member function)

[**test**](https://cplusplus.com/reference/bitset/bitset/test/) - Return bit value (public member function)

s[**et**](https://cplusplus.com/reference/bitset/bitset/set/) - Set bits (public member function)

t[**o\_string**](https://cplusplus.com/reference/bitset/bitset/to_string/) - Convert to string (public member function)

[**to\_ulong**](https://cplusplus.com/reference/bitset/bitset/to_ulong/) - Convert to unsigned long integer (public member function)

**Readings:**

* **Containers:**
  + - **List.docx**
    - **Deque.docx**
    - **Bitset.docx**

#### HW#10:

1. Write a program that contains a “Dog” class whose data consists of int Id and a static int cnt that keeps track of how many dogs have been created. The Dog class should contain default constructor that sets the Id value with the updated vale of cnt and prints a message: cout << “dog #Id:” << X[i].Id. The Dog class should also have a copy constructor that updates cnt and assigns it to the Id# of constructed dog. The copy constructor should output the Id# of the constructed object. The Dog class should have a destructor that reduces the cnt value each time a “Dog” goes out of scope. The destructor should print the Id# of the “dog” that goes out of scope.
   1. Create a **list** array of dogs(size 8).
   2. Output the Id values of all the dogs
   3. After creating the Dog **list**, insert a new dog into location #5.
   4. Explain the stream of outputs that this problem creates.
2. **Write a program that creates a bitset class that contains a 16-bit word that can do the following operations:**
   * **Instantiate bitset S(0xABCD) and bitset Q(0x1248)**
   * **flip all the bits S & Q and print results**
   * **flip bit index N of S & Q and print results**
   * **set bit index N of S & Q and print results**
   * **test bit index N of S and Q**
   * **shift all bits to the right by N bits - operator<<( int N ) & print results**

**Take-home Exam#2 will be handed out – return on Week 13**

# Week: STL Containers - (Tu, Apr 4 – Th, Apr 5(*no classes*) )

**Associative Containers**

* **Sets and Multisets** 
  + Abilities of Sets and Multisets
  + Set and Multiset Operations
  + Examples of Using Sets and Multisets
  + Example of Specifying the Sorting Criterion at Runtime
* **Maps and Multimaps** 
  + Abilities of Maps and Multimaps
  + Map and Multimap Operations
  + Using Maps as Associative Arrays
  + Exception Handling
  + Examples of Using Maps and Multimaps
  + Example with Maps, Strings, and Sorting Criterion at Runtime

**Readings:**

* **Containers:**
  + - **Map.docx**

**HW#11:**

1. Using the **map** container which contains a string for the input word and int cnt for the number of chars in each stored word, write an STL program that takes the following phrases from the Gettysburg Address:

***“Four score and seven years ago our fathers brought forth upon this continent, a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal.***

***Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are met on a great battle-field of that war. We have come to dedicate a portion of that field, as a final resting place for those who here gave their lives that that nation might live. It is altogether fitting and proper that we should do this.***..”

2. Output in ascending order all the words contained in this speech and their corresponding frequency value

* Input a series fruit names (~20) with their corresponding weights in a **multimap** container.
* Print all the fruits and corresponding weights
* Find the heaviest and lighter fruits and print their name and weight

# Week: Spring Recess – (Tu, Apr 11 – Th, Apr 13)

# Week: STL Iterators - (Tu, Apr 18 – Th, Apr 20)

**STL Iterators**

* **Header Files for Iterators**
* **Iterator Categories** 
  + Input Iterators
  + Output Iterators
  + Forward Iterators
  + Bidirectional Iterators
  + Random Access Iterators
  + The Increment and Decrement Problem of Vector Iterators
* **Auxiliary Iterator Functions** 
  + Stepping Iterators Using
  + Processing Iterator Distance Using
  + Swapping Iterator Values Using
* **Iterator Adapters** 
  + Reverse Iterators
  + Insert Iterators
  + Stream Iterators

**Readings:**

**Containers:**

**Overview of Iterators in C++ STL.docx**

**Introduction to Iterators.docx**

**HW#12:**

1.Create integer list of size 24

* + insert elements from -3 to 29
  + using iterator print out all nodes
  + remove all elements divisible by 3
  + using iterator print out all nodes

2.Create integer vector of size 100

* + insert elements from 1 to 100
  + use a forward iterator to iterate over initial half of the vector size and a backward iterator to iterate over the final half of the vector. For each iteration take the sum of the two numbers and print sum.
* **Deadline for submitting Take-Home Exam#2**

# Week - Special Containers - (Tu, Apr 25 – Th, Apr 27)

* **Stack**
  + FILO, push(), pop(), top()
  + Examples using Stacks
* **Queue**
  + FIFO, push(), front(), back(), pop()
  + Examples using Queues
* **Hash** 
  + **Hash Functions**
  + **Collision Resolution**

**Readings:**

**Containers:**

* Stack in STL.docx
* Queue in STL.docx
* Hash Functions in Hash.docx

**HW#13:**

**1.Write a program that uses the STL Stack container that emulates a Reverse Polish Notation calculator. The program should issue an error message if too few values are used with binary operators. For example, “23 - +” should create an error.**

**Take-home Final (Q1, Q2) will be handed out – return on Week 16(May 9)**

# Week - STL Algorithms - (Tu, May 2 - Th, May 4)

**STL Algorithms**

* **Algorithm Header Files**
* **Algorithm Overview**
  + A Brief Introduction
  + Classification of Algorithms
* **Non-Mutating Algorithms** 
  + Counting Elements
  + Minimum and Maximum
  + Searching Elements
* **Mutating Algorithms** 
  + Copying Elements
  + Transforming and Combining Elements
  + Swapping Elements
  + Assigning New Values & Replacing Elements
* **Removing Algorithms** 
  + Removing Certain Values
  + Removing Duplicates

**Readings:**

**Containers:**

* **Summary of Algorithms in STL**

#### GeeksForGeeks:

#### #9 “STL Algorithm Library.docx”

**HW#14:**

* Create an integer vector of 100 elements having random values between 0 and 100. Find the following:
  + find and print all elements having values < 25
  + find and print four consecutive elements with value > 25
* // Given the following source data int source[] = { 1, 4, 4, 6, 1, 2, 2, 3, 1, 6, 6,  6, 5, 7, 5, 4, 4 };
  + initialize the following list coll with elements from source
  + // remove consecutive duplicates and print new list
* Create an integer vector of 100 elements having random values between 0 and 100.
  + Reshuffle the data so that all values < 25 are at the front of the vector

# Week - STL Algorithms - (Tu, May 9 - Th, May 11)

**STL Algorithms (Cont.)**

* **Sorting Algorithms** 
  + Sorting All Elements
  + Partial Sorting.
  + Heap Algorithms
* **Sorted Range Algorithms**
  + Searching Elements
  + Merging Elements
  + Numeric Algorithms
  + Converting Relative and Absolute Values
* **Numeric Algorithms** 
  + Accumulate
  + Inner Product
  + Partial Sum
  + Adjacent Difference
  + Example: The Least Square Method
* **Algorithm Complexity**

 What is Big O notation, and why does it matter

* Formal Definition of Big O notation
* Big O, Little O, Omega & Theta
* Complexity Comparison Between Typical Big Os
* Time & Space Complexity
* Best, Average, Worst, Expected Complexity
* Why Big O doesn’t matter

**Readings:**

* **Containers:**

**Summary of Algorithms in STL**

**HW#15:**

1.Create two vectors of ints (~20) having values between 0 and 10. Calculate their dot product and print result.

2.Calculate their adjacent difference (derivative) of the above two vectors and print the results

# Week – Review Course Material (Tu, May 16)

**Graphs**

* Graph Definitions
* Graph Implementation
* Graph Transversals
* Path Algorithms

**Review**

**Readings**:

1. **Graph Implementation in STL**

**Deadline submission of Take-home Final (Q1, Q2)**