***Classes:***

**Structs VS Class:** A struct, by default, members are public while private in a class. Inheritance in case of a struct public by default, while private in case of a class. Classes can take part in templates. Structs & classes can have associated methods. Classes and structs can have private data members. There can be constructors and destructors in both.

Class’s **copy constructor** can be explicitly invoked; when a class instance is passed to a function, when a class instance is returned from a function

**accumulate**: computes the sum of the given value init and the elements in the range (first,last)

**transform:** applies the given f(x) to a range and stores the result in another range *std::transform( set.begin(),set.end(), set.begin(), ::change(2upper))*

***Variables/Types:***

**C++ map** requires 2 types to declare the container. Cannot respond to [] or .at() (neither does STL list).

**Basic array data structure:** It is of fixed size. Can use generic algorithms on an array via use of pointers. First value is at index 0.

A 4 byte **unsigned int** can hold 8\*4= 32 bits so from 0->2^32 which is 4 billion ish.

**this variable**: a pointer; you can’t change what it pts to; it is bound to calling instance of a method.

Implementing a **Stack** can be done by an array, linked list or a vector.

***Pointers/Iterators:***

**C++ iterators:** act as pointers to a container value, are required to use generic algorithms such as sort or accumulate and are typed specifically to the container they are used with

C++ Pointer byte size depends on the OS

An iterator/pointer to the first element is required to utilize the STL generic sort algorithm on an entire container’s value

**Single Linked Data Structure:** can change its size during execution, is efficient at inserting an element at a particular position, uses pointers to track the next element in the list.

**tail\_->next\_:** tail\_ points to the last node; tail\_ is that node; (\*last).next\_ is what that last node has as its .next\_ value (a ptr); <tail\_.next\_ is a shortcut

***Rules:***

**Big-Oh** notation:

Fastest: O(n•log(n)) Slowest: O(n^n)

**O(1)** constant time **O(log n)** time grows as the log of the size **O(n)** time grows linearly with size **O(n^2)** time squares with growth

**O(2^n)** time grows as the power of size

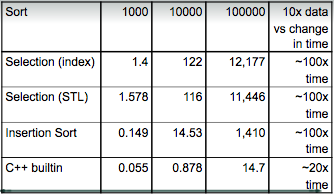
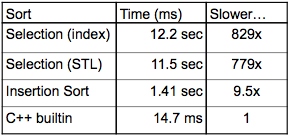
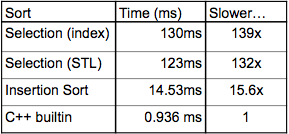
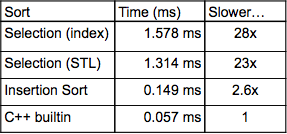
**Rule of 3:** for any object that dynamically allocates memory. In this case you probably define a: cony constructor; define an assign & destructor.

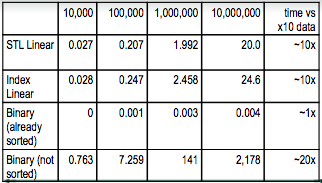
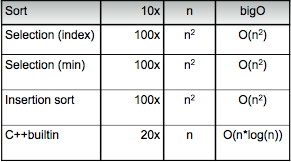
**‘Strongly typed’** C++: every varible must have type.

Class constructor doesn’t have a return type, it initializes an object

Requirement for a properly written **recursive function** is to provide a base case

BIG OH Charts:





***Memory:***

**new**: gets/allocates memory from the OS and returns a pointer to it (single object or array of those objects) new type; new type [size]

**delete**: de-allocates the memory (give back to OS) delete ptr gives ptr back to OS; delete [] ptr deletes all of array element back top OS

**Stack**: used implicitly; think push& pop; grows towards Heap; nay not have ‘holes’; may overflow (run into heap)

**Heap**: used explicitly; think malloc & new; grows towards stack; may have ‘holes’;may OF(run2Stack)

**Leaking Memory**: new some memory; use that memory; reassign the pointer. Since it was never deleted it isn’t owned by the OS nor the user.

