CS 211 : Tues 03/05 (lecture 18)

Prof. Hummel
(he/him)

- **Topics**: **iterators, threaded trees**

MARCH 2024

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

www.a-printable-calendar.com

Notes:

- *Lecture slides available on Canvas*
- *Last week, yay! No HW, but we have a project…*
- ***Project 08** due Friday night (can submit as late as Sunday with late days / penalty).*
- ***Final exam**: Tues 3/12 (noon) or Fri 3/15 (noon)*
  - *You can take exam on either day*
  - *Diagram memory; write a C++ program (with C?)*
  - *Hand-written, no computer, no internet, no notes*

Northwestern University

iterators

- **Recall that iterators are objects that act like pointers**
  - *iterator controls access to the data (e.g. read but not write)*

```
int main()
{
  set<int>  S;

  S.insert(22);
  S.insert(11);
  S.insert(49);
  .
  .

  cout << "Enter an integer> ";
  cin >> x;

  auto iter = s.find(x);
  if (iter == s.end())
    cout << "not found" << endl;
  else
    cout << "found " << *iter << endl;
```

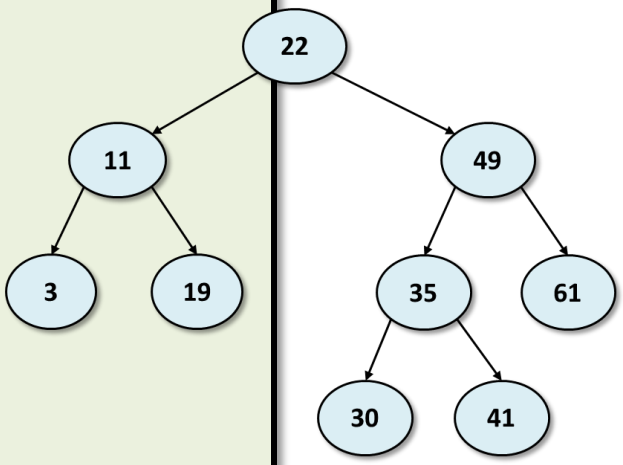
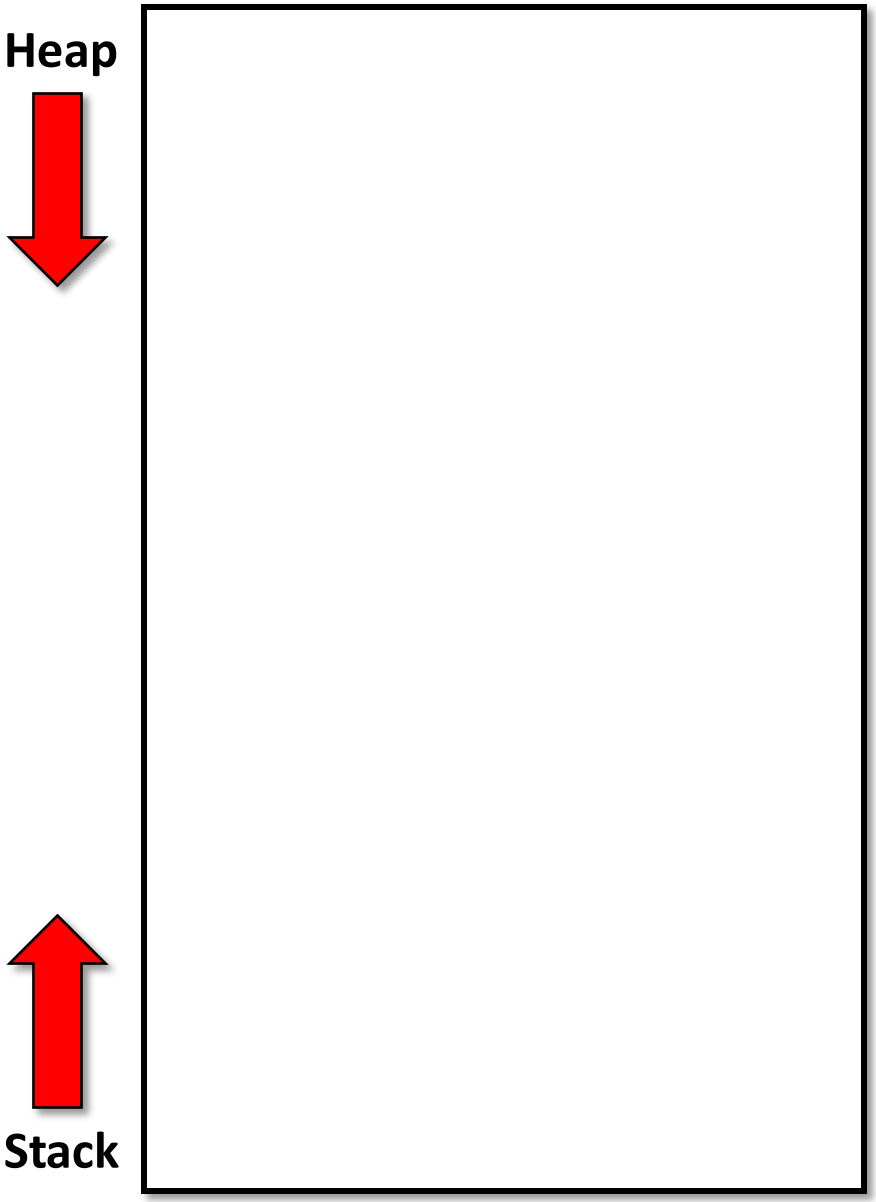


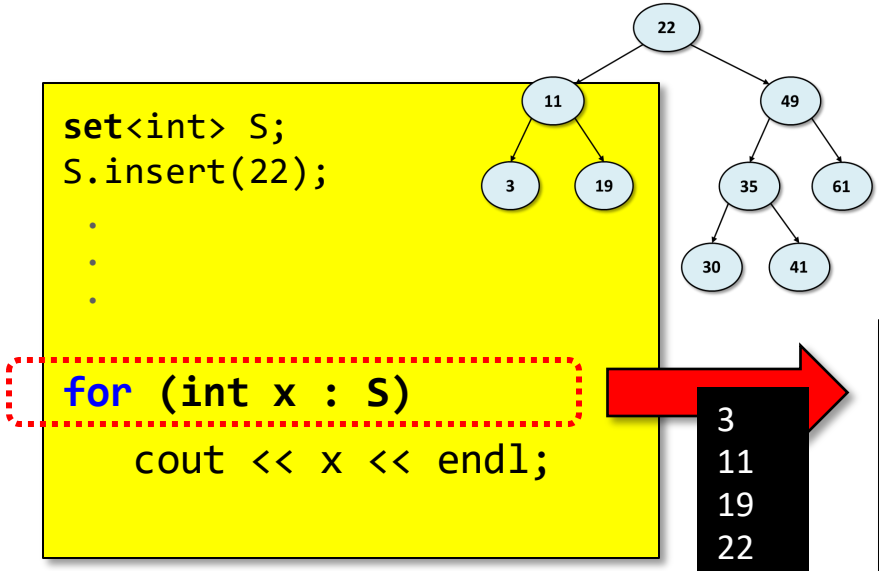
Diagram memory

```
void F(set<int>& S)
{
  auto iter = S.find(20);
  // stop, draw memory
}

int main()
{
  int i;
  double j = 0.0;

  set<int> S;
  S.insert(50);
  S.insert(90);
  S.insert(10);
  S.insert(20);

  F(S);
```



foreach

- C++ supports "foreach" through all the built-in containers
- How?
- Using iterators



Sequence containers

Sequence containers implement data structures which ca

array (C++11)	static contiguous array (class template)
vector	dynamic contiguous array (class template)
deque	double-ended queue (class template)
forward_list (C++11)	singly-linked list (class template)
list	doubly-linked list (class template)

Associative containers

Associative containers implement sorted data structures

set	collection of unique keys, sorted by keys (class template)
map	collection of key-value pairs, sorted by keys, (class template)
multiset	collection of keys, sorted by keys (class template)
multimap	collection of key-value pairs, sorted by keys (class template)

Unordered associative containers

Unordered associative containers implement unsorted (ha amortized, *O(n)* worst-case complexity).

unordered_set (C++11)	collection of unique keys, (class template)
unordered_map (C++11)	collection of key-value pa (class template)
unordered_multiset (C++11)	collection of keys, hashed (class template)
unordered_multimap (C++11)	collection of key-value pa (class template)

Container adaptors

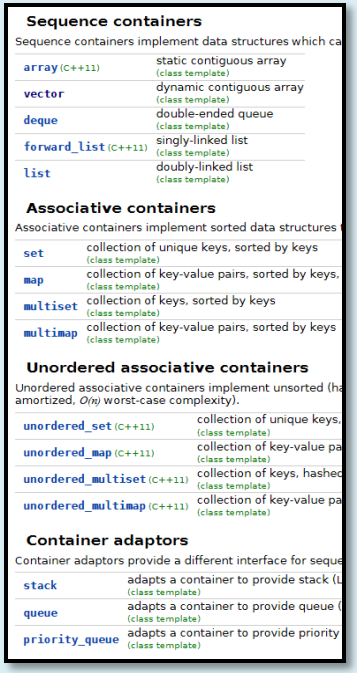
Container adaptors provide a different interface for seque

stack	adapts a container to provide stack (L (class template)
queue	adapts a container to provide queue ((class template)
priority_queue	adapts a container to provide priority (class template)

```
auto iter = S.begin();
while ( iter != S.end() )
{
    int x = *iter;
    cout << x << endl;
    iter++;
}
```

Why?

- **Any container can support foreach by providing three things:**
  1. ***iterator class*** *that overaloads !=, * and ++*
  2. ***begin( )*** *that returns iterator*
  3. ***end( )*** *that returns iterator*



Sequence containers

Sequence containers implement data structures which ca

array (C++11)	static contiguous array (class template)
vector	dynamic contiguous array (class template)
deque	double-ended queue (class template)
forward_list (C++11)	singly-linked list (class template)
list	doubly-linked list (class template)

Associative containers

Associative containers implement sorted data structures

set	collection of unique keys, sorted by keys (class template)
map	collection of key-value pairs, sorted by keys, (class template)
multiset	collection of keys, sorted by keys (class template)
multimap	collection of key-value pairs, sorted by keys (class template)

Unordered associative containers

Unordered associative containers implement unsorted (ha amortized, *O(n)* worst-case complexity).

unordered_set (C++11)	collection of unique keys, (class template)
unordered_map (C++11)	collection of key-value pa (class template)
unordered_multiset (C++11)	collection of keys, hashed (class template)
unordered_multimap (C++11)	collection of key-value pa (class template)

Container adaptors

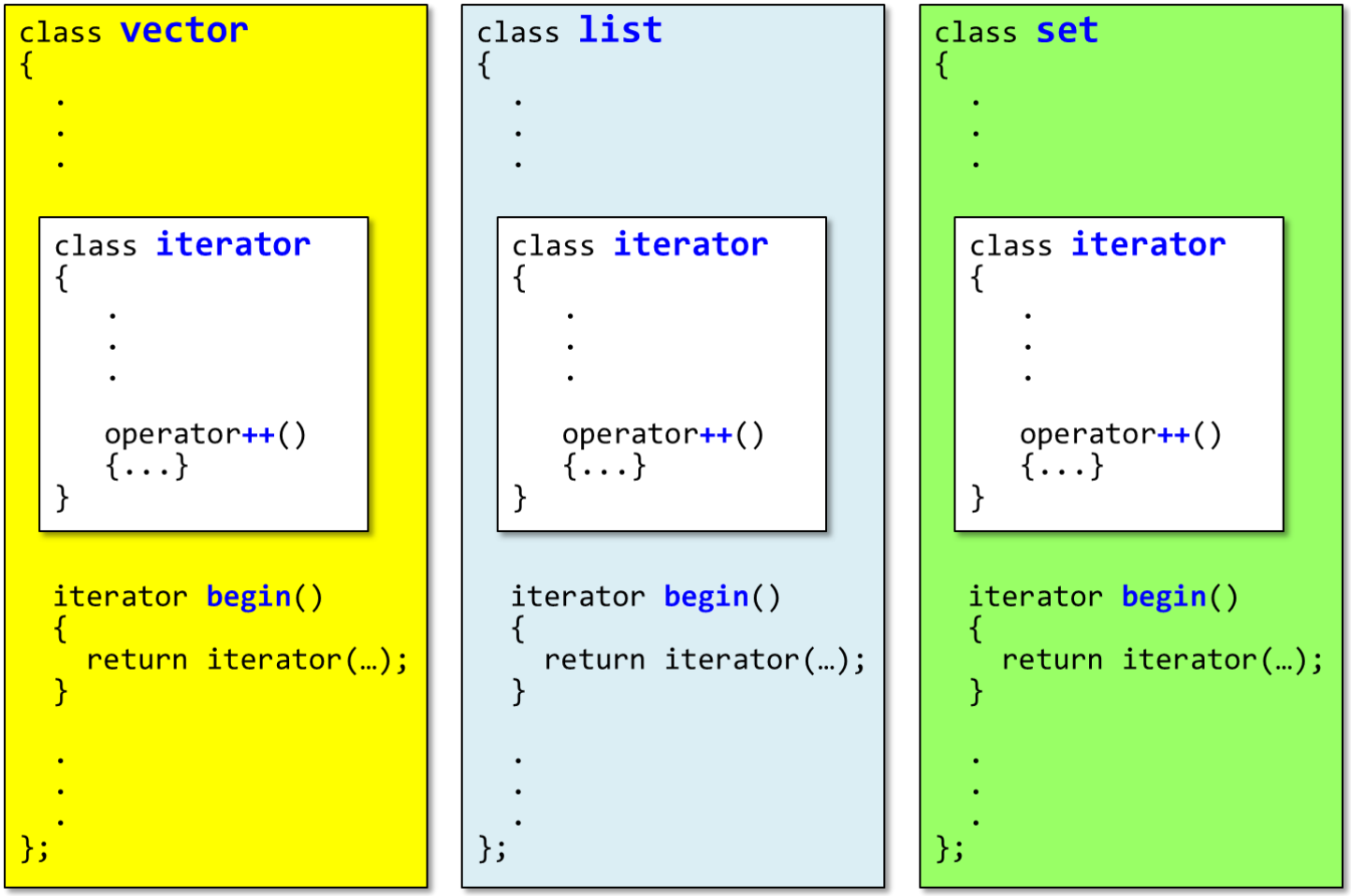
Container adaptors provide a different interface for seque

stack	adapts a container to provide stack (L (class template)
queue	adapts a container to provide queue ((class template)
priority_queue	adapts a container to provide priority (class template)

```
set<string> S;
S.insert(22);
  .
  .
  .
for (int x : S)
    cout << x << endl;
```

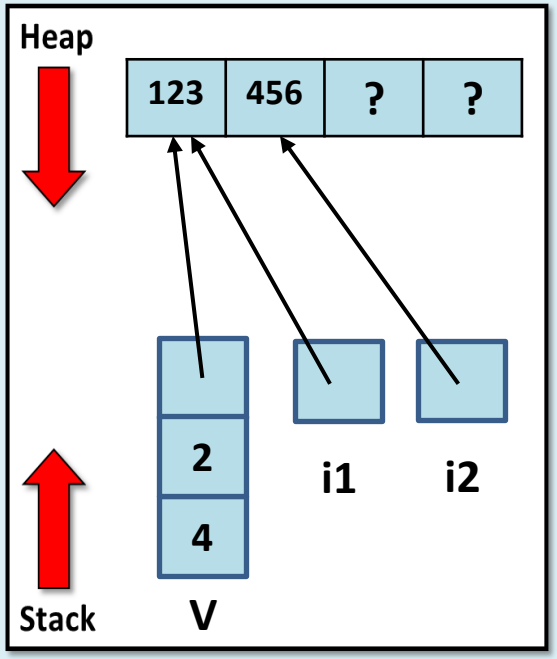
```
auto iter = S.begin();
while ( iter != S.end() )
{
    int x = *iter;
    cout << x << endl;
    iter++;
}
```

class iterator

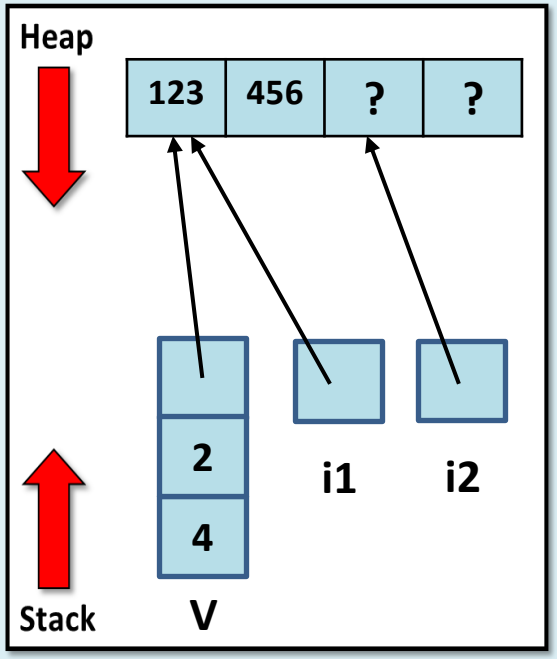


1) ***V*** *is a vector containing 2 elements. Which elements do the iterators* ***V.begin( )*** *and* ***V.end( )*** *denote?*

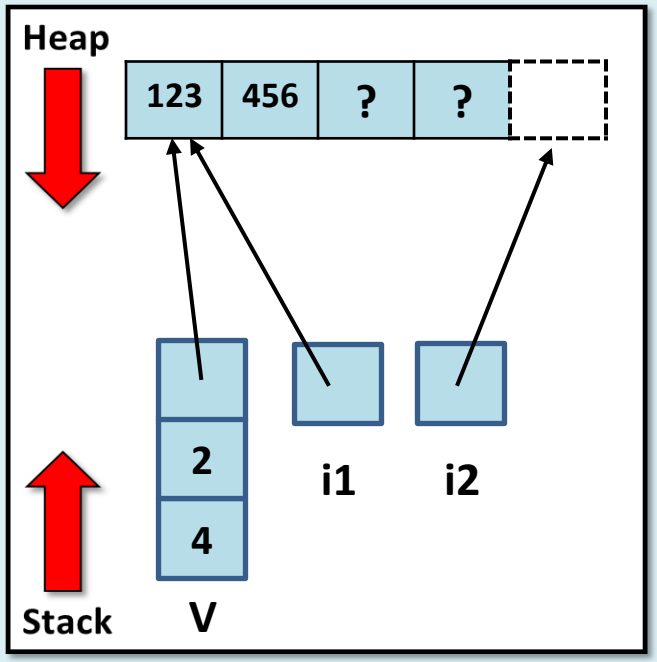
```
int main()
{
  std::vector<int>  V;

  V.push_back(123);
  V.push_back(456);

  auto i1 = V.begin();
  auto i2 = V.end();
```



(A)



(B)



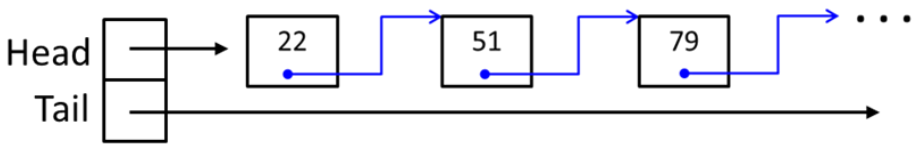
(C)

Replit

- Login to replit.com
- Open team…
- Open project "Lecture 18"

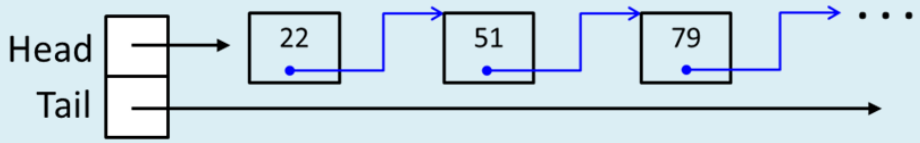
```
template<typename TKey>
class list
{
private:
  NODE *Head;
  NODE *Tail;

  class NODE
  {
    …
  }

  class iterator
  {
  private:
    NODE *Cur;
  public:
    iterator(NODE *head)
      : Cur(head)
    { }

    void operator++()
    { Cur = Cur->Next; }

    …
  }

public:
  .
  .
  .
```

Example: list




```
int main()
{
  list<int> L;

  L.push_back(22);
  L.push_back(51);
  L.push_back(79);
  .
  .
  .

  for (int x : L)
    cout << x << endl;
```

2) For list, what should begin() and end() return?




```
template<typename TKey>
class list
{
public:
 .
 .
 .

 iterator begin() {
  return iterator(this->Head);
 }

 iterator end() {
  return iterator(nullptr);
 }
```

(A)

```
template<typename TKey>
class list
{
public:
 .
 .
 .

 iterator begin() {
  return iterator(this->Cur);
 }

 iterator end() {
  return iterator(nullptr);
 }
```

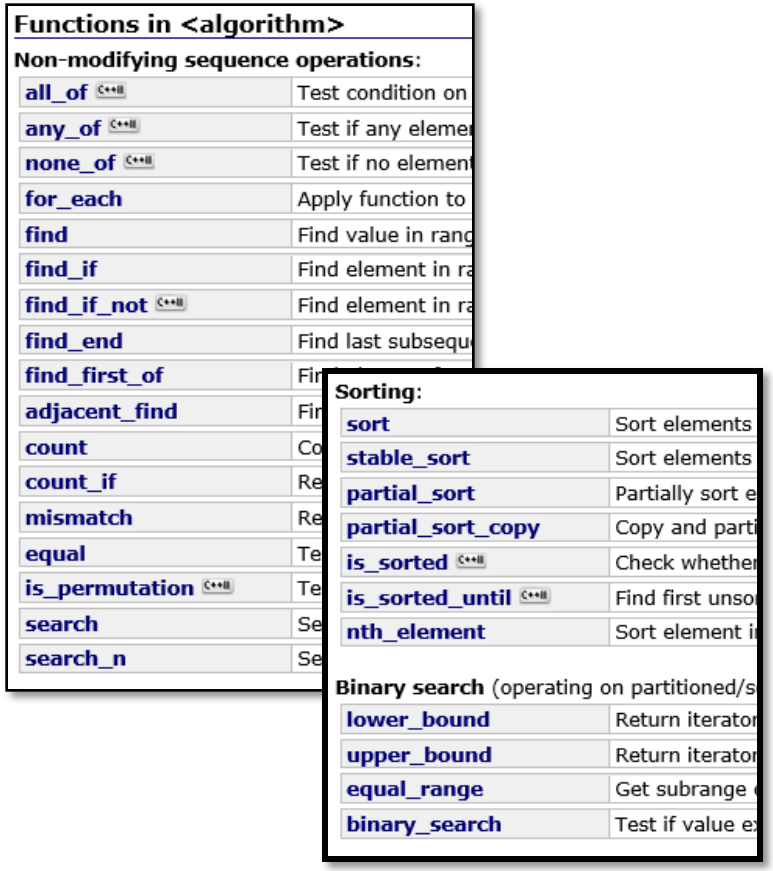
(B)

```
template<typename TKey>
class list
{
public:
 .
 .
 .

 iterator begin() {
  return iterator(this->Head);
 }

 iterator end() {
  return iterator(this->Tail);
 }
```

(C)

std::algorithms

- **Most of the built-in algorithms use iterators to support different types of containers**



Functions in <algorithm>

Non-modifying sequence operations:

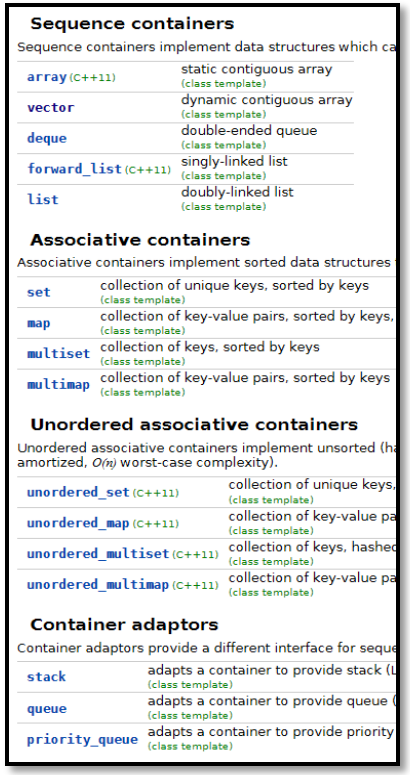
all_of (C++11)	Test condition on
any_of (C++11)	Test if any eleme
none_of (C++11)	Test if no elemen
for_each	Apply function to
find	Find value in rang
find_if	Find element in ra
find_if_not (C++11)	Find element in ra
find_end	Find last subsequ
find_first_of	Fi
adjacent_find	Fi
count	Co
count_if	Re
mismatch	Re
equal	Te
is_permutation (C++11)	Te
search	Se
search_n	Se

Sorting:

sort	Sort elements
stable_sort	Sort elements
partial_sort	Partially sort e
partial_sort_copy	Copy and parti
is_sorted (C++11)	Check whether
is_sorted_until (C++11)	Find first unso
nth_element	Sort element i

Binary search (operating on partitioned/s

lower_bound	Return iterator
upper_bound	Return iterator
equal_range	Get subrange (
binary_search	Test if value e



Sequence containers

Sequence containers implement data structures which ca

array (C++11)	static contiguous array (class template)
vector	dynamic contiguous array (class template)
deque	double-ended queue (class template)
forward_list (C++11)	singly-linked list (class template)
list	doubly-linked list (class template)

Associative containers

Associative containers implement sorted data structures

set	collection of unique keys, sorted by keys (class template)
map	collection of key-value pairs, sorted by keys, (class template)
multiset	collection of keys, sorted by keys (class template)
multimap	collection of key-value pairs, sorted by keys (class template)

Unordered associative containers

Unordered associative containers implement unsorted (ha amortized, *O(n)* worst-case complexity).

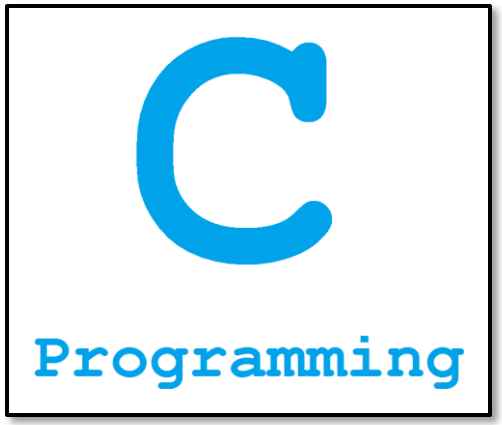
unordered_set (C++11)	collection of unique keys, (class template)
unordered_map (C++11)	collection of key-value pa (class template)
unordered_multiset (C++11)	collection of keys, hashed (class template)
unordered_multimap (C++11)	collection of key-value pa (class template)

Container adaptors

Container adaptors provide a different interface for seque

stack	adapts a container to provide stack (L (class template)
queue	adapts a container to provide queue ((class template)
priority_queue	adapts a container to provide priority (class template)


C++

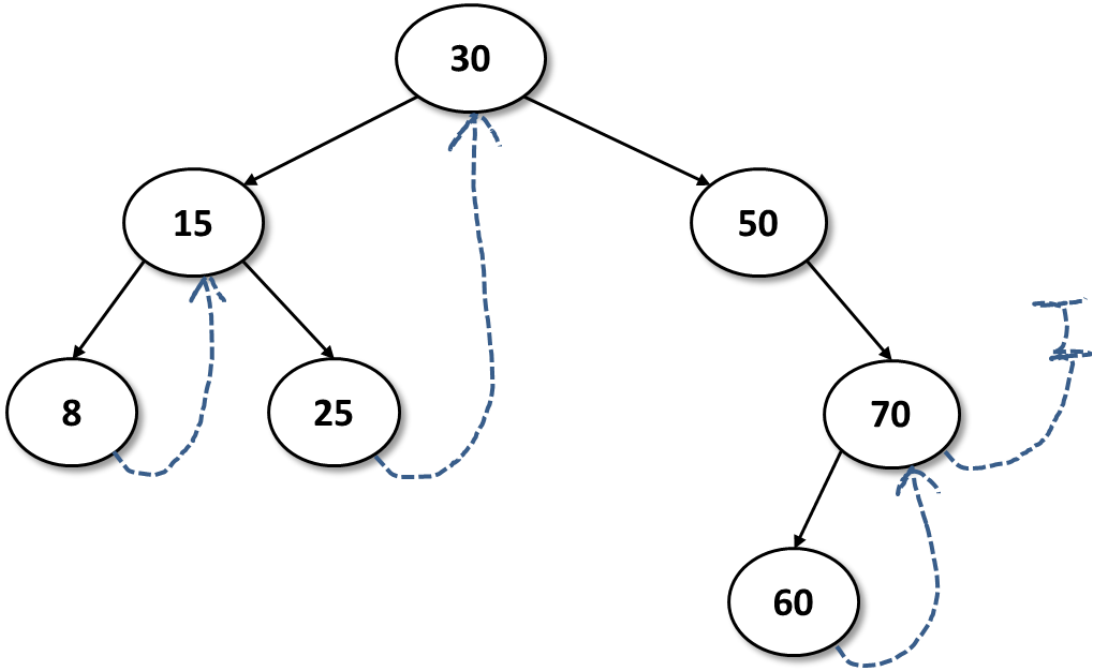

C
Programming

Project 08: set using threaded BST

- Fast --- O(lgN) insert, delete, search
- Efficient inorder traversal --- O(1) space

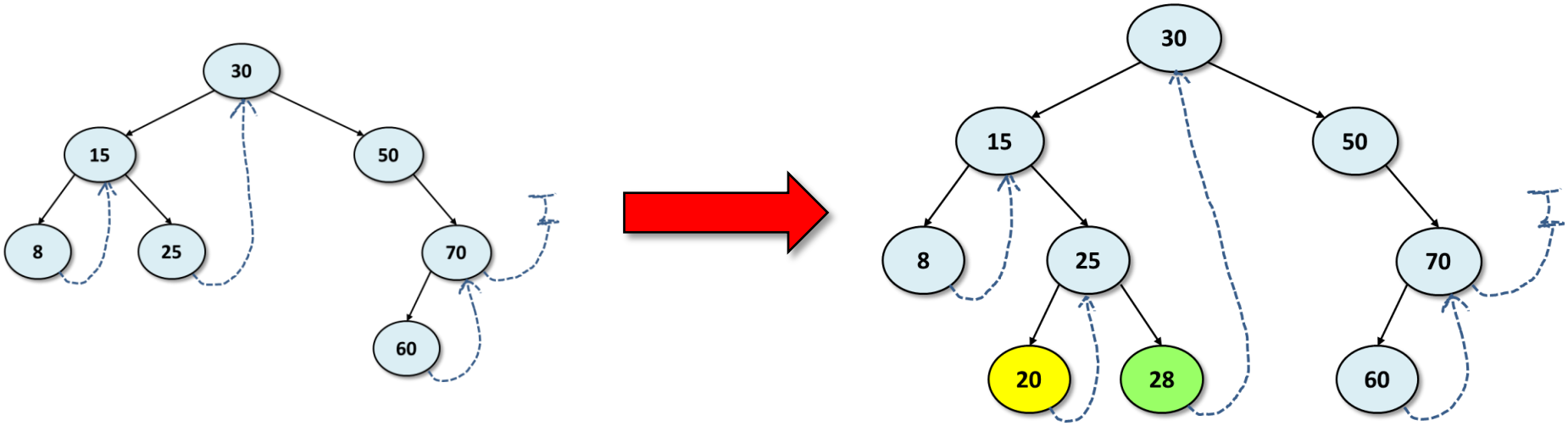
```
set<int> S;
S.insert(30);
S.insert(15);
S.insert(50);
  .
  .
  .

for (int x : S)
    cout << x << endl;
```

```
8
15
25
30
50
60
70
```



Threads

- Threads are created during insert( )
- Threads are used during inorder traversal



Final Exam

- **Here are a couple practice questions…**

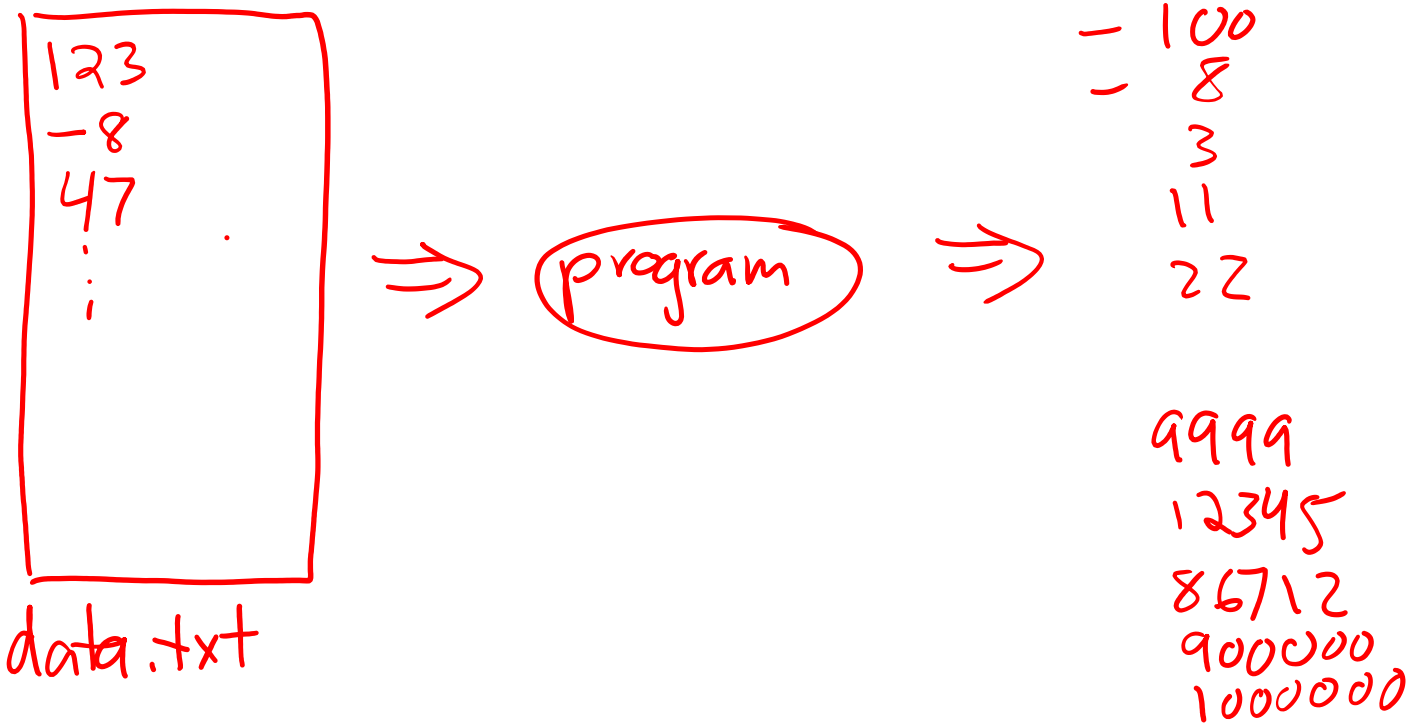
3) ***In front of you is a Linux computer. You type the "ls" command to see what files are given. You see:***

- *main.cpp*
- *util.h*
- *util.cpp*

What command would you type to compile these files and produce a program you can run?

Write a program

- **Write a complete C++ program to input N random integers from a file named "data.txt"; values appear one per line. Output the smallest 5 values and the largest 5 values. You may use any feature of C++ you want; assume N > 10.**



What's due?

Project 08 *is due Friday night*