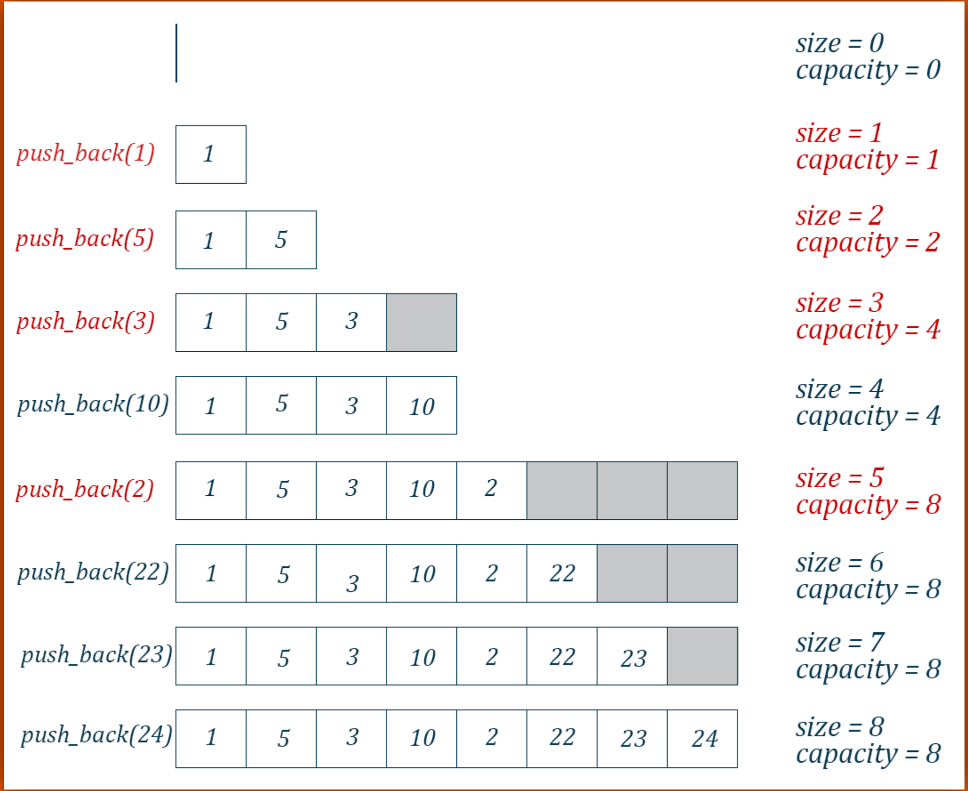
Programming Challenges I

Session 2

CSCI 485/4930 - Spring 2018

C++ Containers

Vector


size = 0
capacity = 0
push_back(1)
1
size = 1
capacity = 1
push_back(5)
1 5
size = 2
capacity = 2
push_back(3)
1 5 3
size = 3
capacity = 4
push_back(10)
1 5 3 10
size = 4
capacity = 4
push_back(2)
1 5 3 10 2
size = 5
capacity = 8
push_back(22)
1 5 3 10 2 22
size = 6
capacity = 8
push_back(23)
1 5 3 10 2 22 23
size = 7
capacity = 8
push_back(24)
1 5 3 10 2 22 23 24
size = 8
capacity = 8

Vectors - push_back(), size(), and operator[]

```
#include <vector>
int main() {
   vector<int> v;                              //create a vector of ints
   v.push_back(10);                            //put values at end of array
   v.push_back(11);
   v.push_back(12);
   v.push_back(13);
   v[0] = 20;                                  //replace with new values
   v[3] = 23;
   for(int j=0; j<v.size(); j++)               //display vector contents
       cout << v[j] << ‘ ‘;                    //20 11 12 23
   cout << endl;
   return 0;
}
```

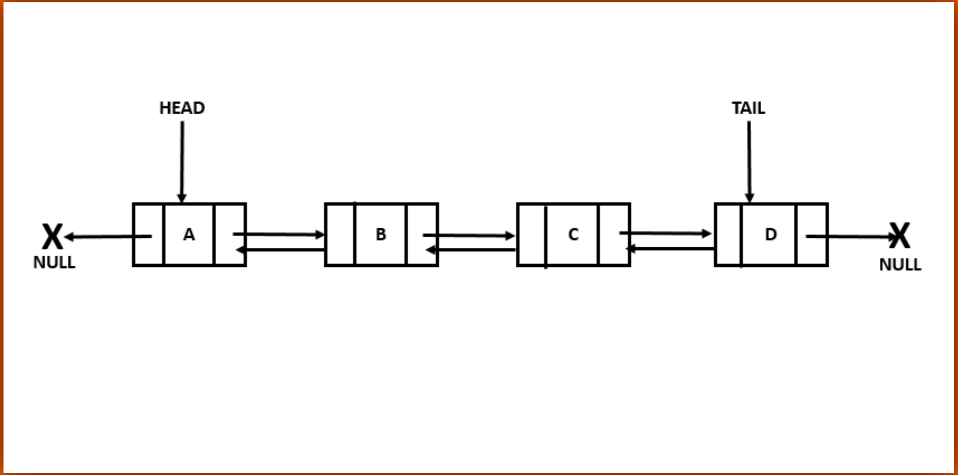
Vectors - swap(), empty(), back(), and pop_back()

```
#include <vector>
int main()  {                                   //an array of doubles
    double arr[] = { 1.1, 2.2, 3.3, 4.4 };
    vector<double> v1(arr, arr+4);             //initialize vector to array
    vector<double> v2(4);                      //empty vector of size 4
    v1.swap(v2);                               //swap contents of v1 and v2
    while( !v2.empty() ){                      //until vector is empty,
        cout << v2.back() << ' ';              //display the last element
        v2.pop_back();                         //remove the last element
    }                                          //output: 4.4 3.3 2.2 1.1
    cout << endl;
    return 0;
}
```

Vectors - insert() and erase()

```
int main() {
   int arr[] = { 100, 110, 120, 130 };   //an array of ints

   vector<int> v(arr, arr+4);              //initialize vector to array

   v.insert( v.begin()+2, 115);            //insert 115 at element 2

   v.erase( v.begin()+2 );                 //erase element 2

   return 0;
}
```

List



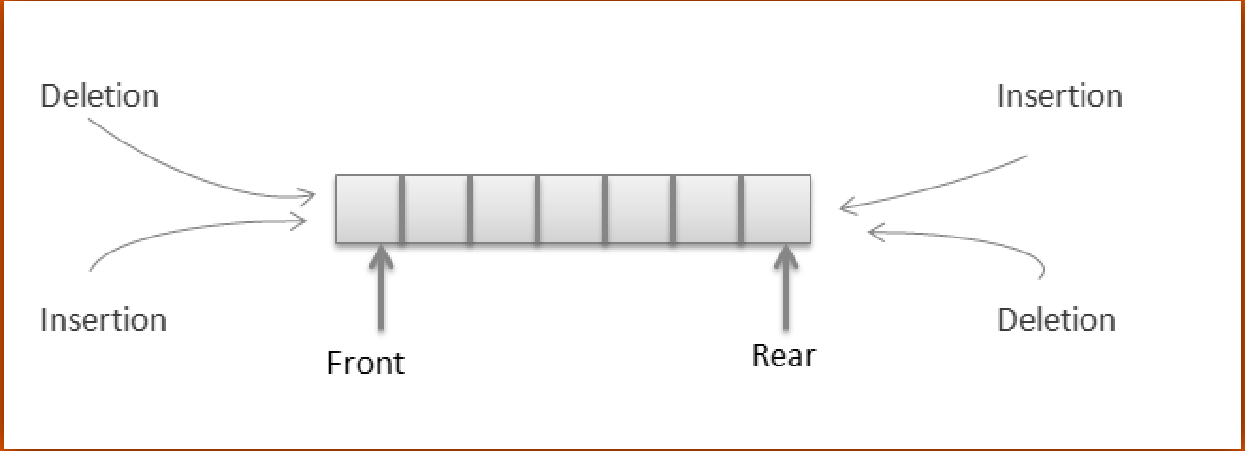
Lists - push_front(), front(), and pop_front()

```
#include <list>
int main() {
   list<int> ilist;
   ilist.push_back(30);                 //push items on back
   ilist.push_back(40);
   ilist.push_front(20);                //push items on front
   ilist.push_front(10);
   int size = ilist.size();             //number of items
   for(int j=0; j<size; j++) {
      cout << ilist.front() << ' ';   //read item from front
      ilist.pop_front();                //pop item off front
   }
   cout << endl;
}
```

Lists - reverse(), merge(), and unique()

```
int main() {
   int j;
   list<int> list1, list2;
   int arr1[] = { 40, 30, 20, 10 };
   int arr2[] = { 15, 20, 25, 30, 35 };
   for(j=0; j<4; j++) list1.push_back( arr1[j] ); //list1: 40, 30, 20, 10
   for(j=0; j<5; j++) list2.push_back( arr2[j] ); //list2: 15, 20, 25, 30, 35
   list1.reverse();                          //reverse list1: 10 20 30 40
   list1.merge(list2);                       //merge list2 into list1
   list1.unique();                           //remove duplicate 20 and 30
   while( !list1.empty() ) {
        cout << list1.front() << ' ';        //read item from front
        list1.pop_front();                   //pop item off front
   }
   cout << endl;
}
```

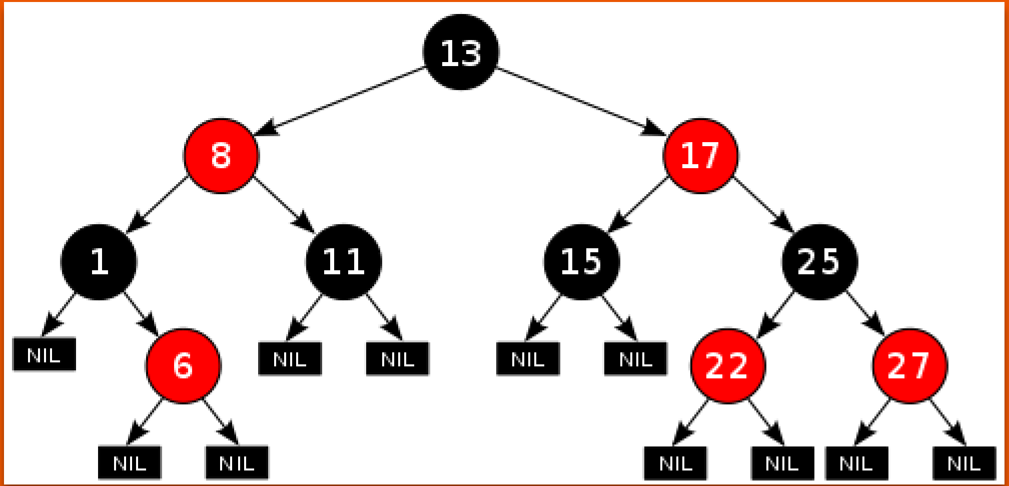
Deque



Deques - push_back(), push_front(), front()

```
#include <deque>
int main() {
    deque<int> deq;
    deq.push_back(30);          //push items on back
    deq.push_back(40);
    deq.push_back(50);
    deq.push_front(20);         //push items on front
    deq.push_front(10);
    deq[2] = 33;                //change middle item
    for(int j=0; j<deq.size(); j++)
        cout << deq[j] << ‘ ’; //display items
    cout << endl;
    return 0;
}
```

Set



Set

```
#include <set>
int main() { //array of string objects
   string names[] = {"Juanita", "Robert","Mary", "Amanda", "Marie"};
    set<string> nameSet(names, names+5); //initialize set to array
    set<string>::iterator iter;              //iterator to set
    nameSet.insert("Yvette");                //insert more names
    nameSet.insert("Larry");
    nameSet.insert("Robert");                //no effect; already in set
    nameSet.insert("Barry");
    nameSet.erase("Mary");                   //erase a name
   cout << "\nSize=" << nameSet.size() << endl; //display size of set
```

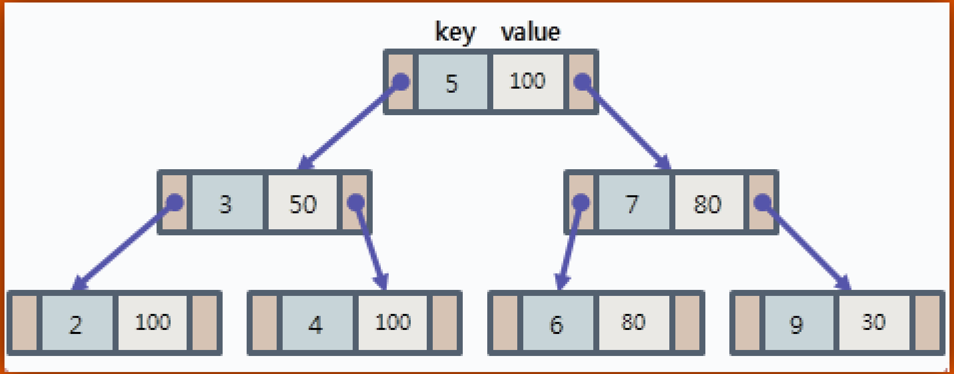
Set

```
    iter = nameSet.begin();                             //display members of set
    while( iter != nameSet.end() )
        cout << *iter++ << '\n';

    string searchName;                                  //get name from user
    cout << "\nEnter name to search for: ";
    cin >> searchName;
    iter = nameSet.find(searchName);                    //find matching name in set
    if( iter == nameSet.end() )
       cout << "The name " << searchName << " is NOT in the set.";
    else
       cout << "The name " << *iter << " IS in the set.";
    cout << endl;
    return 0;
}
```

Set - lower_bound() and upper_bound()

```
int main() {
   set<string > organic;                              //set of string objects
   set<string, less<string> >::iterator iter;    //iterator to set
   organic.insert("Curine");         organic.insert("Xanthine");
   organic.insert("Curarine");       organic.insert("Melamine");
   organic.insert("Cyanimide");      organic.insert("Phenol");
   organic.insert("Aphrodine");      organic.insert("Imidazole");
   organic.insert("Cinchonine");     organic.insert("Palmitamide");
   organic.insert("Cyanimide");
   iter = organic.begin();
   while( iter != organic.end() ) cout << *iter++ << '\n'; //display set
   string lower, upper;
   cin >> lower >> upper;
   iter = organic.lower_bound(lower);
   while( iter != organic.upper_bound(upper) )
      cout << *iter++ << '\n';
}
```

Map



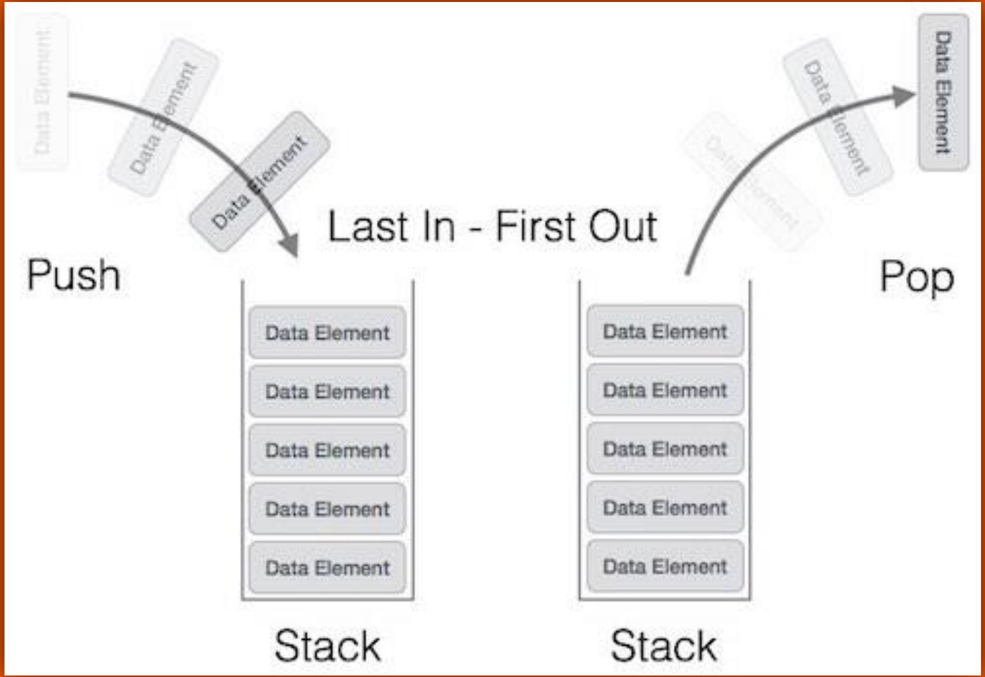
Map

```
#include <map>
int main() {
   string name;
   int pop;
   string states[] = { “Wyoming”, “Colorado”, “Nevada”, “Montana”,
“Arizona”, “Idaho”};
   int pops[] = { 470, 2890, 800, 787, 2718, 944 };
   map<string, int> mapStates;                     //map
   map<string, int>:: iterator iter;               //iterator
   for(int j=0; j<6; j++) {
      name = states[j];                            //get data from arrays
      pop = pops[j];
      mapStates[name] = pop;                       //put it in map
   }
}
```

Map

```
    cout << “Enter state: “;                           //get state from user
    cin >> name;
    pop = mapStates[name];                             //find population
    cout << “Population: “ << pop << “,000\n”;
    cout << endl;                                      //display entire map
    for(iter = mapStates.begin(); iter != mapStates.end(); iter++)
       cout << (*iter).first << ‘ ‘ << (*iter).second << “,\n”;
    return 0;
 }
```

Stack

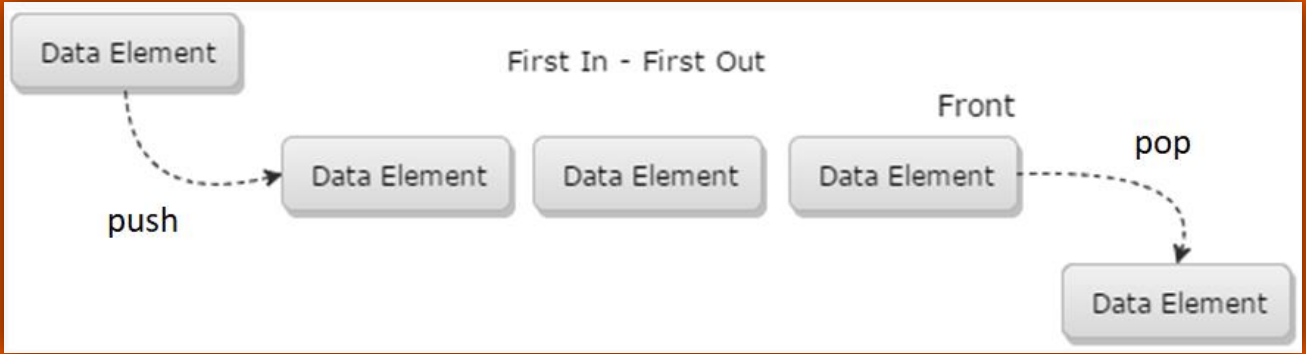

Data Element
Data Element
Data Element
Last In - First Out
Push
Pop
Data Element
Data Element
Data Element
Data Element
Data Element
Data Element
Data Element
Data Element
Data Element
Data Element
Data Element
Data Element
Data Element
Stack
Stack

Stack

Last in first out

Equivalent to deque with operations push_front() and pop_front()

Queue

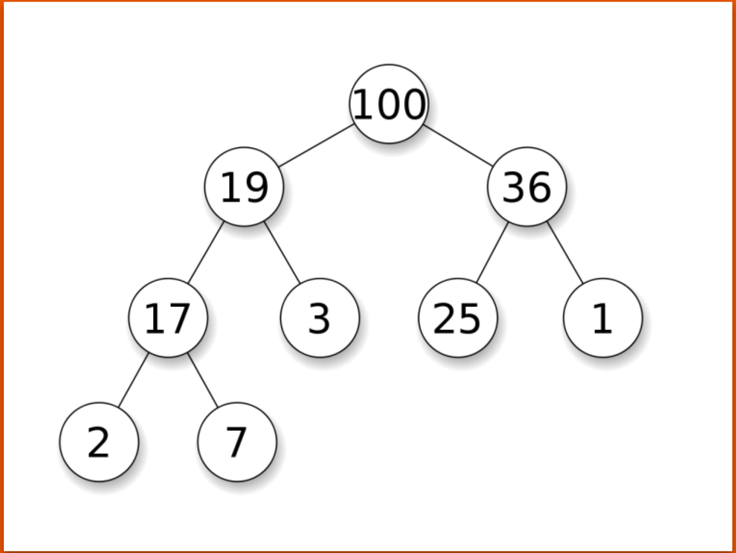

Data Element
First In - First Out
Front
pop
push
Data Element
Data Element
Data Element
Data Element

Queue

First in first out

Equivalent to deque with operations push_back() and pop_front()

Priority Queue



Priority Queue

```
#include <queue>
int main ( ) {
    priority_queue<int> mypq;
    int sum(0);
    for (int i = 1; i <= 10; i++)
        mypq.push(i);
    while (!mypq.empty()) {
        sum += mypq.top();
        mypq.pop();
    }
    cout << "total: " << sum << '\n';
    return 0;
}
```