CSE 333 Section

Section 8
Upcoming Important Dates

- Exercise 7 due 11/20 @ 11:15 AM
- Homework 3 due 11/30 @ 11:59 PM
- Lecture 11/25 Optional Material
- No Class on 11/26 and 11/27
Standard Template Library

- C++’s library for:
  - Containers (familiar data structures)
  - Iterators
  - Algorithms (such as sort and find)
- Stored by value
STL vector

• Analogous to Java’s ArrayList
  – Continuously stored memory
  – Dynamically size
• Located in the <vector> package
• Documentation @
  http://www.cplusplus.com/reference/vector/vector/
STL Iterator

• Each container has an iterator
• Common Operations:
  – Incremented with (++)
  – Copied
  – Copy Constructed
• Additional Operations
  – Dereferencing (LHS/RHS)
  – Decremented with (--)
  – Random Access
Iterating over a vector

```cpp
#include <cstdio>
#include <string>
#include <vector>
#include <iostream>

using namespace std;

int main() {
    vector<int> *vecPtr = new vector<int>();

    for (int i = 0; i <= 10; i++) {
        vecPtr->push_back(i);
    }

    for (vector<int>::iterator it = vecPtr->begin(); it != vecPtr->end(); it++) {
        cout << *it << " ";
    }

    cout << endl;
}
```
Iterating over a vector (in reverse)

```cpp
#include <cstdio>
#include <string>
#include <vector>
#include <iostream>

using namespace std;

int main() {
    vector<int> *vecPtr = new vector<int>();

    for (int i = 0; i <= 10; i++) {
        vecPtr->push_back(i);
    }

    for (vector<int>::reverse_iterator it = vecPtr->rbegin(); it != vecPtr->rend(); it++) {
        cout << *it << " ";
    }

    cout << endl;
}
```
Random Access Vector Iterator

```cpp
#include <iostream>
#include <vector>
#include <cstdlib>

using namespace std;

int main() {
    vector<int> myVector;

    for (int i = 0; i < 10; i++) {
        myVector.push_back(i);
    }

    vector<int>::iterator it = myVector.begin();

    for (int i = 0; i < 10; i++) {
        cout << it[i] << " ";
    }

    cout << endl;
}
```
STL Algorithms

• Functions used on range of elements
• Range = sequence of items accessible through iterators or pointers
• Algorithms operate on values using assignment or copy constructors
• Useful Algorithms:
  – find, count, binary_search, sort, transform, ...
• Documentation @
  http://www.cplusplus.com/reference/algorithm/
# STL Algorithms on vector

```cpp
#include <cstdlib>
#include <string>
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

void print(vector<int> vect) {
    for (vector<int>::iterator it = vect.begin(); it != vect.end(); it++) {
        cout << *it << " ";
    }
    cout << endl;
}

int main() {
    vector<int> vect;

    for (int i = 5; i >= 0; i--)
        vect.push_back(i);

    print(vect);
    sort(vect.begin(), vect.end());
    print(vect);
```
STL Algorithms on vector

```cpp
#include <vector>
#include <iostream>
#include <algorithm>

using namespace std;

int main() {
    vector<int> vect;

    for (int i = 0; i <= 100; i += 2) {
        vect.push_back(i);
    }

    cout << "Has 10? " << binary_search(vect.begin(), vect.end(), 10) << endl;
    cout << "Has 33? " << binary_search(vect.begin(), vect.end(), 33) << endl;
}
```
Section Exercise

• Function Signature:
  vector<int> ProcessQuery(<list<list<string>>> wordLists, list<string> query);

• Input:
  wordLists - A list of lists of words.
  query - A list of words

• Output:
  The index of each word list that contains all the words in query sorted by decreasing matching word count.

(This is similar to the part B6 of Homework 3)