IB9N7 C++ for Quantitative Finance
Worksheet 15

STL2

25 February 2016
(Week 21)

Objectives for this lab session

By the end of this session, you should have completed the following:

- Try out STL!

Exercises

Exercise 1: Set using set and unordered_set

Here we write more classes implementing the interface which we has for the objects `Set1` and `Set2` last week, with two public member functions (i.e. these functions form its interface).

```
void add(int a); //add a to the set
bool isPresent(int a) const; //return whether a is in the set.
```

(a) Create a class `Set3` which implements this interface, by storing all the values which have been added in a `std::set<int>` data member.

(b) Create a class `Set4` which implements this interface, by storing all the values which have been added in a `std::unordered_set<int>` data member.

Exercise 2: Memoization

(a) Implement the following class - by making `calculate` return the factorial of its input. Do not worry about overflow or negative input.

```
class Factorial{
public:
    int calculate(int x); //returns (x!)
};
```

(b) Add a private data member `std::map<int,int> m_memo` which the class uses to remember values of the factorial which it has already calculated. Modify the function `calculate` so that it looks for the answer in `m_memo` if available, and only does a calculation if not. If a calculation is done it will be added to `m_memo` as well as returned. This pattern is called memoization and is a common trick.

Exercise 3: Sort with lambda

Write a function which takes a single `std::vector<std::string>` by reference and sorts it so that the elements are in descending order of length. Use a lambda.

Exercise 4: Lambda capture

Implement the function which is partially shown here. It takes a vector of people's names and ages and returns the number of people with ages greater than 30.

```
int countOver30s(const std::vector<std::tuple<std::string,int>>& v){
    int total = 0;
    // Fill in here. //
    return total;
}
```

You should iterate over `v` using `std::for_each` with a lambda which captures an appropriate local variable.

Exercise 5: Other algorithms

Have a look at the other algorithms available at `http://en.cppreference.com/w/cpp/algorithm`.