CSE 100 Fall 2015

Discussion 2

Wenjia (Wendy) Ouyang

weouyang@eng.ucsd.edu
Announcement

• 1. PA1 submission
  – Oct 6\textsuperscript{th} 8pm: not required
  – Due Oct 8\textsuperscript{th} 8pm: you must submit something before it
  – \textbf{ONLY} perform \textbf{one submission} with both of your student IDs

• 2. Memory leaks
  – We will not be grading on that for PA1
  – It may be graded since PA2

• 3. The same discussion topic for this Fri, next Mon & Wed.
4. Check Tutors Lab hours!

The Autograder queue is now available for CSE 100
1. Standard Template Library (STL)

In Java, the Collections API in package java.util implements standard data structures such as lists, sets, and maps.

C++ has a package that provides similar functionality, namely the Standard Template Library, which is known simply as the STL.
STL features

- The Standard Template Library (STL) is a software library for the C++ programming language that influenced many parts of the C++ Standard Library. It provides four components called:

  **Algorithms**

  **Containers**
  - Sequential: `vector`, `deque`, `list`
  - Associative: `set`, `multiset`, `map`, `multimap`
  - Adapters: `stack`, `queue`, `priority queue`

  **Functional**

  **Iterators**
  - Each container declares a trait for the type of iterator it provides
  - Types: Input, output, forward, bidirectional, & random access
2. STL Container Overview

• A container is a **holder object** that stores a collection of other objects (its elements). They are implemented as **class templates**, which allows a great flexibility in the types supported as elements.

• The container manages the storage space for its elements and provides member functions to access them, either directly or through **iterators**.
STL Vector Sequential Container

• Vectors are sequence containers representing arrays that can change in size (grow and shrink at the end)

• Supports random access iterators

• Similar to—but more powerful than—built-in C/C++ arrays
Vector Member functions

(constructor) Construct vector (public member function )
(destructor) Vector destructor (public member function )
operator= Assign content (public member function )

Iterators:
begin Return iterator to beginning (public member function )
end Return iterator to end (public member function )

Modifiers/Methods:
push_back Add element at the end (public member function )
pop_back Delete last element (public member function )
insert Insert elements (public member function )
erase Erase elements (public member function )
swap Swap content (public member function )

More information see:
http://www.cplusplus.com/reference/vector/vector/
In test_BST.cpp

```cpp
using std::vector;

/**
 * A simple test driver for the BST class template.
 * P1 CSE 100 2012
 * Author: P. Kube (c) 2012
 */
int main() {
  /* Create an STL vector of some ints */
  vector<int> v;
  v.push_back(3);
  v.push_back(4);
  v.push_back(1);
  v.push_back(100);
  v.push_back(-33);
}
```
3. STL Iterator Overview

• **Iterator definitions**

  – An iterator is any object that, **pointing to** some element in a range of elements (such as an array), has the ability to **iterate through** the elements of that range using a set of operators, such as increment (++) and dereference (*) operators.

• Iterators are central to generic programming because they are an interface between containers & algorithms
STL Iterator Categories

- Input and output iterators
- Forward iterators
- Bidirectional iterators
- Random-access iterators
In BSTIterator.hpp

```cpp
template<typename Data>
class BSTIterator : public std::iterator<std::input_iterator_tag, Data> {

private:

    BSTNode<Data>* curr;
```
In `test_BST.cpp`

```cpp
/* Create an STL vector of some ints */
vector<int> v;
v.push_back(3);
v.push_back(4);
v.push_back(1);
v.push_back(100);
v.push_back(-33);

/* Create an instance of BST holding int */
BST<int> b;

/* Insert a few data items. */
vector<int>::iterator vit = v.begin();
vector<int>::iterator ven = v.end();
```
4. PA1

Not start yet?

OMG
What this function do?

```cpp
/** Find the first element of the BST */
static BSTNode<Data>* first(BSTNode<Data>* root) {
  if (root == 0) return nullptr;
  while (root->left != 0) root = root->left;
  return root;
}
```

How could we use it??
Be careful to the syntax!

/** Return an iterator pointing to the first item in the BST. */

```cpp
+ BST.hpp

/** Return an iterator pointing to the first item in the BST (not the root).
   */ // TODO
iterator begin() const {
}

/** Return an iterator pointing past the last item in the BST.
   */
iterator end() const {
    return typename BST<Data>::iterator(0);
}
```

How to return an iterator?
/** Find a Data item in the BST.
 * Return an iterator pointing to the item, or the end
 * iterator if the item is not in the BST.
 */
QUESTIONS?

Good luck for your PA1
Thanks