COP4530 – Data Structures, Algorithms and Generic Programming
Recitation 4
Date: January 27 & 29, 2009

Lab topic:

1) **Take Quiz 4**
2) **Discussion on Assignment 2**

Discussion on Assignment 2.

Your task is to **write 2 template classes** and **rewrite your implementation for *Assignment 1*** to use these two classes. The template classes that you are required to implement are:

1. **The vector template class**: Used to store the flight and number of seats combination.

Part 1: About the Vector Template Class

1. Your task is to build a template **`Vector`** class and name the file **`vector.h`**

2. An object of this class will be used to store the flight and number of seats information read in from the **`FLIGHTS.txt`** file.

3. You are required to implement your own template **`Vector`** class. You CANNOT use *STL vector* objects in your template class to avoid coding for the various required implementations.

4. The class must contain the following implementations:
   a. **Required:**
      i. a default constructor that initializes an array of size 2,
      ii. a destructor,
      iii. a method named **`void push_back(const T &e)`**,
      iv. the **`[ ]`** operator,
      v. the method **`int size() const`**

   b. **Optional (make private if not implemented):**
      i. Copy Constructor
      ii. Assignment operator

   c. **Any additional methods or operator overloads needed.**

5. A sample class declaration of the **`vector.h`** file in your implementation could look similar to the one below. Notice that the class in encapsulated in the namespace **`blah`** to more clearly distinguish the class from the *STL vector* class. However, using namespaces in this manner is optional.

```
#ifndef MYVECTOR_H
#define MYVECTOR_H

#include <iostream>
#include <stdlib.h>    // EXIT_FAILURE, size_t

namespace blah
{

   template <typename T>
   class Vector;

   //---------------------------------
   //     Vector<T>
   //---------------------------------

   template <typename T>
   class Vector
   {
   public:
     // constructors - specify size and an initial value
     Vector  ();
     ~Vector ();

     // member operators
     T&          operator [] (int) const;

     // other methods
     int   size        () const;
     int   capacity    () const;

     // Container class protocol
     int      push_back    (const T&);

     void dump       (std::ostream& os) const;

   protected:
     // data
     int size, capacity;
     T*  content;   // pointer to the primative array elements

   };
} //end of namespace blah

#endif
```

6. Brief description of each method/operator overloads:

   a. **`Vector ():`**
      i. The **`size`** is initialized to 0 since we do not have any elements in a newly declared vector.
      ii. The **`capacity`** is initialized to 2 since the project requirement states that the default constructor "*initializes an array of size 2*".
      iii. The array (named **`content`** in our example) is initialized to a size of 2.

   b. **`~Vector():`**
      i. Deallocate the dynamically allocated memory for the array.
      ii. Deallocate any other dynamically allocated memory
      iii. Set **`size`** and **`capacity`** to 0.

   c. **`T& operator [] (int ind):`**
      i. Check the bounds for the index **`ind`** that is passed in. If the index is invalid, print out an error message.
      ii. If the index is valid, return the value of the element located at the index **`ind`** of the array.

   d. **`int size() const:`**
      i. Returns the size of the array.

   e. **`int capacity() const:`**
      i. Returns the capacity of the array.
      ii. This method is optional.

   f. **`void push_back(const T&):`**
      i. Check to see if there is currently enough space to add T. If there is, just add T to the array
      ii. If there isn't enough space, reallocate memory for a larger array. You may do so by **doubling** the capacity of the array. Copy the contents over to the new larger array and then add T to the array.

   g. **`void dump(std::ostream &os ) const:`**
      i. Prints out the contents of the array.
      ii. This method is optional.

References

Topic	Links
STL `vector`	**1.** http://www.sgi.com/tech/stl/Vector.html
STL `list`	1. http://www.sgi.com/tech/stl/List.html