CS180 Homework 4

Due via email on 3/23 by 11:59pm

Note: You should actually code and test these functions! I expect full C++ code, which will compile and run when I test them.

1. As you may have noticed, the vector class in C++ has much less functionality than the list class in Python. In as effort to "upgrade" the vector class we wrote, you will write the following 3 functions described below in our Vector.h file. (Please download Vector.h from the course schedule webpage and insert your functions into this class, then submit the entire Vector.h file via email by the due date.)

   You must also submit a file with a main function that tests these functions.

```
  /** Determines the number of occurrences of given value.
 *  Input parameter:  The value
 *  Returns the number of occurrences
 */
int count(const ItemType& val) const;

/** Determines the first index at which give value occurs (if any).
 *  Input Parameter:  The value
 *  Returns the first index, if found;  otherwise returns size (clearly an invalid index)
 */
int index(const ItemType& val) const;

  /** Reverse the contents of the vector.
 */
void reverse();
```

2. We provided an our own implementation of a list class that mimics the interface of std::list. However, std::list has additional behaviors that were not in our implementation. Two of these are the following:

```
  /** Reverse the order of elements in the list.  All iterators
    * remain valid and continue to point to the same elements.
    * This function is linear time.
    **/
  void reverse();

  /** Remove elements with specific value
   *
   * Removes from the list all the elements with a specific value.
   * This  reduces the list size by the amount of elements removed.
   **/
  void remove ( const ItemType& value );
```

   Give a valid implementation of these methods in the context of our list implementation. Again, you must download the List.h file and insert these function. You are also required to submit a test file which contains a main that tests these functions.

   Please note in reverse that to maintain the validity of all existing iterators, you must not create or destroy any nodes nor change the element of a node. The only way to successfully implement this behavior is by relinking the existing nodes into the desired order.

Please note that in remove, unlike the function erase which erases elements by their position (iterator), this function (remove) removes elements by their value. (Don't forget to delete the nodes when they are removed, also!)

3. Extra Credit:

   Another method supported by the std::list class is the following.

```
/** Splice one list into another.
  * All elements of other list are removed from that list and
  * inserted into this list immediately before the given position.
  *
  * A call runs in constant time and all iterators remain valid,
  * including iterators that point to elements of other.
  *
  * @param position must be an iterator into this list
  * @param other must be a distinct list (i.e., &other != this)
  */
void splice(iterator position, list& other);
```

   Give a valid implementation of the slice method, implemented in our list class.