CS180 Homework 3

Due via email on 10/13 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. (Taken from C-5.3) Rewrite the function erase from our implementation of vectors so that if the number of elements gets below $N/4$, you shrink the array size by half.

   In our code (under Vector.h on the schedule page), this means if numItems gets below currentCapacity divided by 4, then currentCapacity should be divided by 2 and all the elements copied into a new array of the appropriate size.

2. We coded most of the functions from the list class in our List.h. However, there is one notable exception: erase. The following is the declaration for the erase function:

```
/**  Removes from the list container a single element (position).
 * This effectively reduces the list size by one, and calls the destructor
 * for that element.
 * Parameter position: an iterator pointing to a single element in the list
 *       that will be removed
 * Returns an iterator to the element following the element that was erased.
 **/
iterator erase (iterator position)();
```

   Give a valid implementation of the erase method in the context of our list implementation, List.h, available on the schedule of our course webpage.

3. 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. One of these is 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();
```

   Give a valid implementation of the reverse method in the context of our list implementation. Please note 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.

4. 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.