**Question 1:**   
  
In this homework, you are going to implement your own version of vectors as an ADT to manage a collection of doubles using dynamic arrays. (Note that vector is implemented as a template, but you are only required to implement it as a class.) Specifically, your Vector\_double ADT should include the following member functions:

* a default constructor.
* the "Big-3" (destructor, copy constructor, assignment opt).
* the "capacity" function: Returns the number of elements that the vector could store without allocating more storage.
* the "empty" function: Tests if the vector container is empty.
* the "erase" function: Removes an element or a range of elements in a vector from the specified positions.
* the "insert" function: Inserts an element or a number of elements into the vector at a specified position.
* the "pop\_back" function: Deletes the element at the end of the vector.
* the "push\_back" function": Adds an element to the end of the vector.
* the "resize" function: Specifies a new size for a vector. The new size can be smaller or larger than the old size.
* the "shrink\_to\_fit" function: Discards excess capacity.
* the "size" function" Returns the number of elements stored in the vector.
* the overloaded output operator (<<) to print out the elements in a vector.

Please refer to [this webpage](http://msdn.microsoft.com/en-us/library/9xd04bzs.aspx) for further details and sample of usage about most of the above functions in the built-in vector class.   
  
  
  
**Question 2:**Implement an singly linked list ADT to store a collection of integers. (You are allowed to extend the demo code posted on iLearn under Week #14 for this homework). Make sure you test these new functions in the main() function. Your ADT will include the following member functions:

* a default constructor
* the "big-3"
* a member function that inserts a number at the front of the list
* a member function that deletes a number from the list, where this number can be located anywhere on the list.
* a member function that checks whether an integer is on the list.
* a member function that returns the size of the list.
* an overloaded put operator (<<) to print out all the data items stored on a linked list. Note that you are recommended to overload this operator as a friend function of the List class.
* a member function that reverses a linked list without recreating a temporary copy of this linked list. In other words, your function CAN NOT use the 'new' operator. Here is an example, if a list contains the following data items, 3 -> 5 -> 1 -> 7; this reverse() function will change the list to 7 -> 1 -> 5 -> 3.

**Submission status**

|  |  |
| --- | --- |
| Submission status | No attempt |
| Grading status | Not graded |
| Due date | Friday, 8 May 2015, 11:55 PM |