**Linked List Members**

T& back() – Returns the last node in the list.

iterator begin() – Returns an iterator which references the first node in the list.

void clear() – Removes all nodes in the list.

bool empty() – Returns true if the list has no nodes, and false otherwise

iterator end() – Returns an iterator which references the end of the list.

iterator erase() – removes the node referenced by the iterator and moves the iterator to the next node.

iterator erase(iterator first, iterator last) – removes the nodes between (and including) the two iterators.

T& front() – returns the first node in the list.

void insert(iterator i, int n, const T& el) – inserts n copies of el before the iterator.

void insert(iterator i, iterator first, iterator last) – inserts the elements between first and last before i.

list() – constructor for an empty list.

list(int I, const T& el = T()) – constructor for list with n copies of el.

list(iterator first, iterator last) – constructor for list with elements in the range between first and last.

list(const list<T>& lst) – copy constructor.

int max\_size() – returns the maximum number of nodes in the list.

void merge(list<T>& lst) – add the 2nd list to the first list, in sorted order.

void pop\_back() – remove the last node in the list.

void pop\_front() – remove the first node in the list.

void push\_back(const T& el) – add el to the end of the list.

void push\_front(const T& el) – add el to the front of the list.

void remove(const T& el) – remove all nodes that include el.

void remove\_if(Pred f) – remove all nodes that satisfy function f().

void resize(int n, const T& el=T()) – resizes the list, if necessary using the constructor to create new nodes.

void reverse() – reverses the list.

reverse\_iterator rbegin() – returns an iterator to the last element.

reverse\_iterator rend() – returns an iterator to the start of the list.

int size() – returns the number of nodes in the list.

void sort() – sorts the list.

void splice(iterator I, list<T>& lst) – removes nodes of list lst, inserting them into the list before i.

void swap(list<T>& lst) – swap the contents of the list with lst.

void unique() – remove duplicates from the list.