Simon Chen
N10013388
sc4900
HW 7

1)

- draw pictures showing how the links change (or don't change) for programming questions
- provide the pseudo code for the methods

Initial List - > where h is header, n is nullptr

|h | → |1| → |2| → |3| → n

(a) The copy constructor, List(const List & rhs). This method must take O(n) time.

|h | → |1| → |2| → |3| → n

Go through the list, copying the values into the list on the left side.

(b) The destructor, ∼List( ). This method must take O(n) time.

|h | →n

Go through the list, and use the delete function to delete the node from the heap

(c) The method front( ). It performs as stated in www.cplusplus.com/reference/forward_list/forward_list/front/ This method must take O(1) time.

|h | → |1| → |2| → |3| → n

Just return a reference to the node after the header, in this case the node that contains value 1.

(d) The method merge(List & alist). It performs as stated in www.cplusplus.com/reference/forward_list/forward_list/merge/

alist = |h | → |1| → |4| → |6| → n

list becomes:

|h | → |1| → |1| → |2| → |3| → |4| → |6| → n

Go through both lists, checking which one is lower than the other one and putting that one into the first list. If it's not lower, go to the next item in the list unitl it is. If it reaches a nullptr, then insert the entire list.

(e) The method remove adjacent duplicates( ). The method removes any element if it is adjacent to a node containing the same item 2 . Thus if the list contained a, a, a, b, b, c, c, c, c afterwards it would contain a, b, c. If the list contains a, b, a, b, a then afterwards it contains a, b, a, b, a (i.e. no change, since no items adjacent to each other were the same).

list is:

|h | → |1| → |1| → |2| → |3| → |4| → |6| → n

list becomes:
|h | → |1| → |2| → |3| → |4| → |6| → n

Check if the next node is the same if it is, remove it and repeat until it isn't the same. Then move onto the next node.

(f) The method remove if(Predicate pred) that performs as stated in www.cplusplus.com/reference/forward_list/forward_list/remove_if/ This method must run in O(n) time. Your method should call your method erase after.

list is:
|h | → |1| → |2| → |3| → |4| → |6| → n

pred: is greater than 2

list becomes:
|h | → |1| → |2| → n

It checks if the pred is true, if it is then remove the node and continue through the list checking each node with the pred.

2)

f represents location of int first

```
Stack<char> s;   f[   |   |   |   ]
s.push('a') :       [ fa |   |   |   ]
s.push('b') :       [ a | fb |  |   ]
s.push('d') :       [ a | b |  fd  |   ]
s.pop()             [ a | fb  |      |   ]
s.push('c') :       [ a | b |  fc  |   ]
s.pop() :           [ a | fb |    |   ]
s.pop() :           [ fa |  |    |   ]
```

3)

n represents nullptr

```
Stack<char> s;  n
s.push('a') :       |a | → n
s.push('b') :       |b| → |a | → n
s.push('d') :       |d| → |b| → |a | → n
s.pop()             |b| → |a | → n
s.push('c') :       |c| → |b| → |a | → n
s.pop() :           |b| → |a | → n
s.pop() :           |a | → n
```

4)

n represents nullptr, h represents head, t represents tail

```
Queue<char> q;   h,t,n
q.enqueue('a');   |h,ta|  →  n
q.enqueue('b');   |h a| → |t b| →  n
q.enqueue('d');  |h a| → |b| → |t d| →n
q.dequeue();       |h b| → |t d| →n
q.enqueue('c');   |h b| → |t d| → |c| →n
q.dequeue();       |h d| → |t c| →n
q.dequeue();       |h t c| →n
```

5)

f represents pter to first, b represents pter to b

```
Queue<char> q;   [f   |   |   |   b]
q.enqueue('a');    [f,b a  |   |   |   ]
q.enqueue('b');   [f a  | b b  |   |   ]
q.enqueue('c');   [f a  |  b  |b c  |   ]
q.dequeue();       [  |f  b  |b c  |   ]
q.enqueue('d');  [  |f  b  | c  |b  d ]
q.dequeue();      [  |  |f c  |b  d ]
q.dequeue();      [  |  | |f,b  d ]
q.enqueue('e');  [b e  |  | |f  d ]
q.enqueue('f');  [ e  |b  f  | |f  d ]
q.enqueue('g');  [ e  |  f  |b g  |f  d ]
q.enqueue('h');   [f d  |  e  | f  | g  |b  h   |   |    |   ]
```