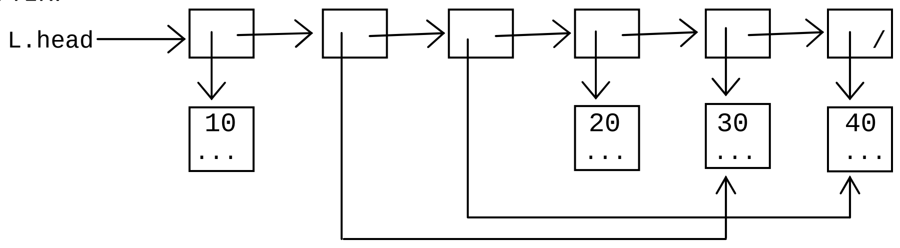
CSS 342 Exam Name________________________________

Be clear. You don't have to comment code, but use meaningful names and proper indentation. Show all work!!! Don't erase.
On the actual exam, space was left for problem solutions.

1. Give the complexity of each of the following. Assume n items. No proof is needed. (10 pts)

(a). One insert of the insertion sort of lab1	O(n)
(b). The operator+ of lab2 (the union of two IntSets)	O(n)
(c). The remove of lab2 (assuming it does not resize)	O(1)
(d). A binary search for one item in a sorted array	O(log n)
(e). Bubble sort on a linked list	O(n^2)

2. Assume a linked list has been built as displayed. Draw the links and memory that show the status of the list after the following function is executed. Show all work on the exam. Although your links, etc. should be correct on the existing list, when you're done, redraw the list. (10 pts)
AFTER:



3. There is no copy constructor in the Rational class (int numerator and denominator data members), yet it works correctly, e.g., `Rational x(1,2), y(x);` Why? (5 pts)

There is always a default copy constructor which does a memberwise copy. This is sufficient for the int data members.

4. Consider the following class. … (15 pts)

(a). Set the time for object **clock1** to be 2 hours and 15 minutes ahead of **clock2**. The objects have been instantiated and are not necessarily equal.

```
clock1.setHours(clock2.getHours());
clock1.setMinutes(clock2.getMinutes());
clock1.setSeconds(clock2.getSeconds());
clock1.advanceTime(2,15,0);
```

(b). Print the difference in time (only hours, minutes) between **clock1** and **clock2** (assume clock1 is ahead of clock2 on the same day). Make sure you output non-negative values for the minutes.

```
int diffHours, diffMinutes;
diffHours = clock1.getHours() – clock2.getHours();
diffMinutes = clock1.getMinutes() – clock2.getMinutes();
if (diffMinutes < 0) {
   diffHours--;
   diffMinutes += 60;
}
cout << diffHours << “Hours, “ << diffMinutes << “Minutes” << endl;
```

5. Suppose T(n) = O(n log n). Define what this means. (5 pts)

There exists constants c, k > 0 so that $T(n) \le c \cdot n \log n$ for all $n > k$.

6. Find the complexity (tight big-oh bound) for the following code. Show summations for "for" loops. Show all work.

(15 pts)

$$T(n) = O(\sum_{i=1}^{n} \underbrace{\sum_{j=i}^{i*n} 1} + \underbrace{\sum_{i=0}^{3n-1} 1}_{=3n})$$

$$\downarrow$$

$$\sum_{j=1}^{i*n} 1 - \sum_{j=1}^{i-1} 1 = i \cdot n - (i-1) = i \cdot n - i + 1$$

$$= O(3n + \sum_{i=1}^{n} (i \cdot n - i + 1)$$

$$= O(3n + n\sum_{i=1}^{n} i - \sum_{i=1}^{n} i + \sum_{i=1}^{n} 1)$$

$$= O(3n + n(n(n+1))/2 - n(n+1)/2 + n)$$

$$= O(3n + \tfrac{1}{2} n^3 + \tfrac{1}{2} n^2 - \tfrac{1}{2} n^2 - \tfrac{1}{2} n + n) = O(n^3)$$

7. Consider your IntSet class of lab2:

(15 pts)

```
    class IntSet {
       ...
    public:
       IntSet(...);              // takes up to 5 int parameters as items in the set
       ...                       // destructor, copy constructor, operator+, etc.
       void insertDoubleMax(const IntSet&);  // inserts double the max element of param
       ...
    private:
       bool* inSet;   // array element is true if subscript is in the set, is never NULL
       int size;      // size of the array, e.g., size is 10, subscripts are 0 to 9
    };


void IntSet::insertDoubleMax(const IntSet& rhs) {
   int value = 2*(rhs.size-1);
   if (value < size) {
      inSet[value] = true;
      return;
   }

   bool* temp = new bool[value+1];
   int i;
   for (i=0; i < size; i++)
      temp[i] = inSet[i];
   for (; i < value; i++)
      temp[i] = false;

   temp[value] = true;
   delete [] inSet;
   inSet = temp;
   temp = NULL;
   size = value+1;
}
```

```
8.  class List {     // lab3, no index array                                    (25 pts)
    public:
       List ();
       ~List ();
       . . .
       void moveToEnd(...);
    private:
       struct Node {
          Employee* data;
          Node* next;
       };
       Node* head;
    };
```

Write a new member function for a List class called `moveToEnd` which will find and move the parameter item to the end of the list. If it is not in the list, or it is already at the end of the list, do nothing.

You must handle all situations. **Any code you use, you must write.** You must actually move the node (move pointers around), not just swap information. (Note that after the move, the list is no longer sorted.) Assume a fully implemented Employee class.

For example:

```
Employee emp(40);
a.moveToEnd(emp);
```

before:

```
           _____    _____    _____    _____    _____    _____    _____    _____
a.head-->|__|_-|->|__|_-|->|__|_-|->|__|_-|->|__|_-|->|__|_-|->|__|_-|->|__|_/|
          ↓        ↓        ↓        ↓        ↓        ↓        ↓        ↓
          20       30       40       50       60       70       80       90
```

after:

```
           _____    _____    _____    _____    _____    _____    _____    _____
a.head-->|__|_-|->|__|_-|->|__|_-|->|__|_-|->|__|_-|->|__|_-|->|__|_-|->|__|_/|
          ↓        ↓        ↓        ↓        ↓        ↓        ↓        ↓
          20       30       50       60       70       80       90       40
```

```
void List::moveToEnd(const Employee& target) {
   if (isEmpty() || head->next == NULL)
      return;

   Node* current;                           // set to the node to move
   Node* previous;                          // used for unlinking, walk to end
   bool found = false;
   if (*head->data == target) {
      current = head;
      head = previous = head->next;
      found = true;
   }
   else {
      previous = head;
      current = head->next;
      while (current->next != NULL && target < *current->data && !found) {
         if (target == *current->data) {
            previous->next = current->next;
            found = true;
         }
         else {
            previous = current;
            current = current->next;
         }
      }
   }

   if (found) {
      while (previous->next != NULL) {
         previous = previous->next;
      }
      previous->next = current;
      current->next = NULL;
      current = previous = NULL;
}
```