1. Recall the public interfaces of classes List and ListIterator:

```cpp
typedef int ListItemType;

class ListIterator{
    public:
        const ListItemType& operator*(); // dereference the iterator
        ListIterator operator++(); // prefix ++
        bool operator==(const ListIterator& rhs) const;
        bool operator!=(const ListIterator& rhs) const;
        friend class List;
        /* private members here ... */
    };

class List{
    public:
        List(); List(const List& aList); // constructors
        ~List(); // destructor
        bool isEmpty() const;
        bool getLength() const;
        ListIterator insert(ListIterator iter, ListItemType newItem);
        // inserts item at spot before iter, returns an iterator
        // to the newly inserted item
        void retrieve(ListIterator iter, ListItemType& dataItem) const;
        // retrieves the item pointed to by iter, places it in
        // dataItem
        ListIterator remove(ListIterator iter);
        // removes the item pointed to by iter from the list and
        // returns an iterator to the next item
        ListIterator begin() const;
        // iterator to first item of list
        ListIterator end() const;
        // iterator value to test whether an iterator has reached
        // the end of the list
        /* private members here ... */
    };
```

Data Structures, Sample Test Questions for the Material after Test 2, with Answers

1. Recall the public interfaces of classes List and ListIterator:
Implement the following problems using iterators. Your code should work in a non-member function of the class List. In other words, you may not access the private data of the class List.

(a) Write a few lines of C++ code which use iterators to print out all the elements of the list aList.

(b) Write a few lines of C++ code which use iterators to change all the elements of the list aList by adding 1 to each of them (so an initial list of 3,6,7,9,0 will become 4,7,8,10,1).

(c) Write a few lines of C++ code which use iterators to delete the even integers from the list aList (so an initial list of 3,6,7,9,0 will become 3,7,9).

2. Consider the following list of words:

apple, tree, car, dog, yellow, frog, gun, harp

(a) Alphabetize the above list using an insertion sort. Show your work.

(b) Alphabetize the above list using a bubble sort. Show your work. How many complete passes are necessary for the bubble sort to ensure the list is sorted?

(c) Alphabetize the above list using a merge sort. Show your work.

(d) Consider an initially empty binary search tree (BST). Place each of the above words into the BST in the order given above. (Use alphabetical order to make your comparisons.) Draw the completed binary search tree.

3. Consider the following binary tree:

```
    G
   / \   
  D   T
 / \   / \  
B   M   X
 / \ / \  \  
A  C I  K  Y
   \  
     P
```

(a) What is the result of a postorder traversal of the above tree?

(b) What is the result of an inorder traversal of the above tree?

(c) Is the above tree a binary search tree? Why or why not?

4. Consider the following list of integers:
5, 54, 125, 105, 25, 104, 20, 100, 50, 159

(a) Slightly modify the integers above so that it is appropriate to perform a radix sort on the above list.

(b) Perform a radix sort on the above list. Show your work.

5. What is the order of each of the following tasks? (Choose from $O(1)$, $O(\log_2 n)$, $O(n)$, $O(n \log_2 n)$, $O(n^2)$, $O(2^n)$; each order may appear more than once.)

(a) Popping an item off a stack containing $n$ items.

(b) Performing a Towers of Hanoi algorithm with $n$ disks.

(c) Using quicksort to sort an array of $n$ integers, in the average case.

(d) Using quicksort to sort an array of $n$ integers, in the worst case.

(e) Inserting a single item into a binary search tree containing $n$ items, in the average case.

(f) Performing a bubble sort on an array of $n$ integers, in the worst case.

(g) Displaying all $n$ elements in a sorted linked list.

(h) Performing a binary search of a sorted array of $n$ strings, in the worst case.

6. Perform a quicksort on the following list of integers. Show your work. Make sure you specify what happens with the pivot at each step.

0, 15, 7, 27, 4, 5

7. Recall the private data of classes TreeNode and Tree:

```cpp
typedef string TreeItemType;
class TreeNode{
    private: TreeItemType item;
    TreeNode* leftChildPtr;
    TreeNode* rightChildPtr;
    /* function members here... */
};
class Tree{
    private: TreeNode* root;
    /* function members here... */
};
```

(a) Write the pseudocode of a preorder traversal of a binary tree.

(b) Write the C++ function definition for a Tree member function

    ```cpp
    void Tree::preorderTraverse() const;
    ```
which prints out all the elements of the tree via a preorder traversal. You should also implement any auxiliary functions needed to make \texttt{preorderTraverse} work.
Answers

1. Recall the public interfaces of classes List and ListIterator:

typedef int ListItemType;
class ListIterator{
   public:
      const ListItemType& operator*(); // dereference the iterator
      ListIterator operator++();     // prefix ++
      bool operator==(const ListIterator& rhs) const;
      bool operator!=(const ListIterator& rhs) const;
      friend class List;
      /* private members here ... */
   };

class List{
   public:
      List(); List(const List& aList); // constructors
      ~List();                       // destructor
      bool isEmpty() const;
      bool getLength() const;
      ListIterator insert(ListIterator iter, ListItemType newItem);
          // inserts item at spot before iter, returns an iterator
          // to the newly inserted item
      void retrieve(ListIterator iter, ListItemType& dataItem) const;
          // retrieves the item pointed to by iter, places it in
          // dataItem
      ListIterator remove(ListIterator iter);
          // removes the item pointed to by iter from the list and
          // returns an iterator to the next item
      ListIterator begin() const;
          // iterator to first item of list
      ListIterator end() const;
          // iterator value to test whether an iterator has reached
          // the end of the list
      /* private members here ... */
   };

Implement the following problems using iterators. Your code should work in a non-member function of the class List. In other words, you may not access the private data of the class List.

(a) Write a few lines of C++ code which use iterators to print out all the elements of the list aList.

**Solution:**

```cpp
for (ListIterator iter = aList.begin();
    iter != aList.end(); ++iter)
    cout << *iter << endl;
```

(b) Write a few lines of C++ code which use iterators to change all the elements of the list aList by adding 1 to each of them (so an initial list of 3,6,7,9,0 will become 4,7,8,10,1).

**Solution:**

```cpp
ListIterator iter = aList.begin();
while(iter != aList.end()){
    ListItemType item = *iter;
    iter = aList.remove(iter); // removes item, increments iter
    aList.insert(iter, item+1); // places item+1 before current value
    // of iter
}
```

(c) Write a few lines of C++ code which use iterators to delete the even integers from the list aList (so an initial list of 3,6,7,9,0 will become 3,7,9).

**Solution:**

```cpp
ListIterator iter = aList.begin();
while(iter != aList.end()){
    ListItemType item = *iter;
    if (item%2 == 0)
        iter = aList.remove(iter); // removes item, increments iter
    else
        ++iter; // merely increment iter
}
```

2. Consider the following list of words:

apple, tree, car, dog, yellow, frog, gun, harp

(a) Alphabetize the above list using an insertion sort. Show your work.

**Solution:**

```
original: apple, tree, car, dog, yellow, frog, gun, harp
1st pass: apple, tree, car, dog, yellow, frog, gun, harp
2nd pass: apple, car, tree, dog, yellow, frog, gun, harp
```
3rd pass: apple, car, dog, tree, yellow, frog, gun, harp
4th pass: apple, car, dog, tree, yellow, frog, gun, harp
5th pass: apple, car, dog, frog, tree, yellow, gun, harp
6th pass: apple, car, dog, frog, gun, tree, yellow, harp
7th pass: apple, car, dog, frog, gun, harp, tree, yellow

(b) Alphabetize the above list using a bubble sort. Show your work. How
many complete passes are necessary for the bubble sort to ensure the
list is sorted?

Solution:

original: apple, tree, car, dog, yellow, frog, gun, harp
1st pass: apple, tree, car, dog, yellow, frog, gun, harp
apple, car, tree, dog, yellow, frog, gun, harp
apple, car, dog, tree, yellow, frog, gun, harp
apple, car, dog, tree, yellow, frog, gun, harp
apple, car, dog, tree, frog, yellow, gun, harp
apple, car, dog, tree, frog, gun, yellow, harp
apple, car, dog, tree, frog, gun, harp, yellow
apple, car, dog, tree, frog, gun, harp, yellow

(After 1st full pass, we know “yellow” is in the correct place.)

2nd pass: apple, car, dog, tree, frog, gun, harp, yellow
apple, car, dog, tree, frog, gun, harp, yellow
apple, car, dog, tree, frog, gun, harp, yellow
apple, car, dog, frog, tree, gun, harp, yellow
apple, car, dog, frog, gun, tree, harp, yellow
apple, car, dog, frog, gun, harp, tree, yellow

(After 2nd full pass, we know “tree, yellow” in correct place.) The
third pass, (from “apple” to “harp” only) will verify that the list is
already sorted, and so bubble sort can exit after 3 full passes.

(c) Alphabetize the above list using a merge sort. Show your work.

```plaintext
apple, tree, car, dog, yellow, frog, gun, harp
  /    
apple, tree, car, dog             yellow, frog, gun, harp
  /    
apple, tree, car, dog yellow, frog             gun, harp
  /    
apple, tree, car, dog yellow frog, gun, harp
  /    
apple, tree, car, dog frog, yellow gun, harp
  /    
apple, tree, car, dog frog, gun, harp, yellow
  /    
apple, car, dog, tree frog, gun, harp, yellow
  /    
apple, car, dog, frog, gun, harp, tree, yellow
```
(d) Consider an initially empty binary search tree (BST). Place each of the above words into the BST in the order given above. (Use alphabetical order to make your comparisons.) Draw the completed binary search tree.

Solution:

```
    apple
     \  
      tree
       /  
      car  yellow
        \    
         dog
          \   
           frog
            \ 
             gun
              \ 
               harp
```

3. Consider the following binary tree:

```
    G
     /  
    D   T
     /  
    B   M   X
       /  
      A   C   I   K   Y
         \   
          P
```

(a) What is the result of a postorder traversal of the above tree?

Solution: ACBDIPKMYXTG

(b) What is the result of an inorder traversal of the above tree?

Solution: ABCDGIMKPTXY

(c) Is the above tree a binary search tree? Why or why not?

Solution: No. The inorder traversal is not in alphabetical order (since the M comes before the K).

4. Consider the following list of integers:

```
5, 54, 125, 105, 25, 104, 20, 100, 50, 159
```

(a) Slightly modify the integers above so that it is appropriate to perform a radix sort on the above list.
Solution: Radix sort only applies to integers that have the same number of digits (or strings that have the same number of characters). For these integers, we fill in initial 0s so that each integer has 3 digits:

005, 054, 125, 105, 025, 104, 020, 100, 050, 159

(b) Perform a radix sort on the above list. Show your work.

Solution:
Last digit: 0 group: 020, 100, 050; 4 group: 054, 104; 5 group: 005, 125, 105, 025; 9 group: 159
Recombine: 020, 100, 050, 054, 104, 125, 105, 025, 159

2nd digit: 0 group: 100, 104, 005, 105; 2 group: 020, 125, 025; 5 group: 050, 054, 159
Recombine: 100, 104, 005, 105, 020, 125, 025, 050, 054, 159

1st digit: 0 group: 005, 020, 025, 050, 054; 1 group: 100, 104, 105, 125, 159
Recombine: 005, 020, 025, 050, 054, 100, 104, 105, 125, 159

5. What is the order of each of the following tasks? (Choose from $O(1)$, $O(\log_2 n)$, $O(n)$, $O(n \log_2 n)$, $O(n^2)$, $O(2^n)$; each order may appear more than once.)

(a) Popping an item off a stack containing $n$ items.
   Solution: $O(1)$.

(b) Performing a Towers of Hanoi algorithm with $n$ disks.
   Solution: $O(2^n)$.

(c) Using quicksort to sort an array of $n$ integers, in the average case.
   Solution: $O(n \log_2 n)$.

(d) Using quicksort to sort an array of $n$ integers, in the worst case.
   Solution: $O(n^2)$.

(e) Inserting a single item into a binary search tree containing $n$ items, in the average case.
   Solution: $O(\log_2 n)$.

(f) Performing a bubble sort on an array of $n$ integers, in the worst case.
   Solution: $O(n^2)$.

(g) Displaying all $n$ elements in a sorted linked list.
   Solution: $O(n)$.

(h) Performing a binary search of a sorted array of $n$ strings, in the worst case.
   Solution: $O(\log_2 n)$. 
6. Perform a quicksort on the following list of integers. Show your work. Make sure you specify what happens with the pivot at each step.

0, 15, 7, 27, 4, 5

**Solution:** P means pivot, and the swaps are indicated beneath.

\[
\begin{array}{cccccc}
0 & 15 & 7 & 27 & 4 & 5 \\
(P) \\
\end{array}
\]

The first pivot is 0: all the other elements are greater than the pivot, so the list is unchanged in the first pass. But we now know that the pivot 0 is in the correct place: the left subarray is empty, and so is already sorted (base case), now the next step is to sort the right subarray 15 7 27 4 5:

\[
\begin{array}{cccccc}
0 & 15 & 7 & 27 & 4 & 5 \\
(P) \\
\end{array}
\]

Here are the swaps for the pivot 15:

\[
\begin{array}{cccccc}
0 & 15 & 7 & 27 & 4 & 5 \\
(P) & \text{//swap//} \\
0 & 15 & 7 & 4 & 27 & 5 \\
(P) & \text{//swap//} \\
0 & 15 & 7 & 4 & 5 & 27 \\
(P) \\
\end{array}
\]

Now swap the pivot 15:

\[
\begin{array}{cccccc}
0 & 15 & 7 & 4 & 5 & 27 \\
\text{//---swap---//} \\
0 & 5 & 7 & 4 & 15 & 27 \\
(P) \\
\end{array}
\]

Now the left subarray is 5 7 4 (recall 0 was already done as a previous pivot). The right subarray is 27, which we know is already sorted since it has only one element. So to sort the left subarray,

\[
\begin{array}{cccccc}
0 & 5 & 7 & 4 & 15 & 27 \\
(P) \\
\end{array}
\]

Now we must swap

\[
\begin{array}{cccccc}
0 & 5 & 7 & 4 & 15 & 27 \\
(P) & \text{//swap//} \\
0 & 5 & 4 & 7 & 15 & 27 \\
(P) \\
\end{array}
\]
Now swap the pivot.

0 5 4 7 15 27
\-swap-/
0 4 5 7 15 27
(P)

At this point, the left subarray 4 and right subarray 7 both have only one element, so we know they’re sorted. Quicksort is finished.

7. Recall the private data of classes TreeNode and Tree:

```cpp
typedef string TreeItemType;
class TreeNode{
    private: TreeItemType item;
    TreeNode* leftChildPtr;
    TreeNode* rightChildPtr;
    /* function members here... */
};
class Tree{
    private: TreeNode* root;
    /* function members here...*/
};
```

(a) Write the pseudocode of a preorder traversal of a binary tree.

**Solution:**

```cpp
preorder(tree node){
    if (tree node is not empty){ // not base case
        visit node
        preorder (left child of node)
        preorder (right child of node)
    }
}
```

(b) Write the C++ function definition for a Tree member function

```cpp
void Tree::preorderTraverse() const;
```

which prints out all the elements of the tree via a preorder traversal. You should also implement any auxiliary functions needed to make `preorderTraverse` work.

**Solution:**

```cpp
void Tree::preorder(TreeNode* treePtr) const{
    if (treePtr != NULL){
        cout << treePtr->item << endl;
        preorder(treePtr->leftChildPtr);
        preorder(treePtr->rightChildPtr);
    }
}
void Tree::preorderTraverse() const{
    preorder(root);
}