4-26 Final Exam In-Lab Possible Questions

Sunday, April 26, 2020 4:04 PM

In-lab Final Exam Ideas:

Following are the possible topics for the in-lab final exam for next week. These all assume that we have:

```
struct Node {
   int data;
   Node *pNext;
};
```

and in main we have:

```
Node *pHead = NULL;  // pointer to the head of the list
Node *pTemp;         // Used to get new nodes
Node *pHeadA = NULL;
Node *pHeadB = NULL;
```

We will also provide for you the code that is used to create the list and the code to traverse and display the list elements. You can also assume that lists being processed already exist, and that the pointers for new lists are initialized to NULL. For the examples below, unless specified otherwise assume the list we are starting with is the list:

```
2->3->5->6->8->9->11
```

<u>You may work on these ahead of time and have electronic copies that you will use during lab, but you may not collaborate with anyone else.</u> Do not post any solutions on Piazza or anywhere else online. You may reuse code we have developed during class. We will likely compare all submitted solutions against each other for similarity, as we have done for programming projects.

The in-lab test questions will be similar or the same as one of the following:

1. Split a list at the node with a designated value. The function header would be:

   ```
   splitList( Node *pOriginal, int splitValue, Node *&pHeadA, Node *&pHeadB)
   ```

   Assume the list we are starting with is the list: 2->3->5->6->8->9->11

   Calling `splitList(…)` from within main would look like:

   ```
   cout << "Enter value for split: ";
   cin >> userInput;
   splitList(pHead, userInput, pHeadA, pHeadB);
   displayList( pHeadB);
   displayList( pHeadA);
   ```

   and it would result in the following output:

   ```
   Enter value for split: 6
   8->9->11
   2->3->5->6
   ```

2. Traverse a list, creating two new lists: one with all the even values, and the other with all the odd values. The function header would be:

   ```
   evenOddLists( Node *pOriginal, Node *&pHeadEven, Node *&pHeadOdd)
   ```

   Assume the list we are starting with is the list: 2->3->5->6->8->9->11

   Calling `evenOddLists(…)` from within main would look like the following:

   ```
   evenOddLists( pHead, pHeadA, pHeadB);
   displayList( pHeadA);
   displayList( pHeadB);
   ```

   and it would result in the following output:

   ```
   2->6->8
   3->5->9->11
   ```

3. Delete all the list nodes that contain even numbers. The function header would be:

   ```
   deleteEvenNumbers( Node *&pHead)
   ```

   Assume the list we are starting with is the list: 2->3->5->6->8->9->11

   Calling `deleteEvenNumbers(…)` from within main would look like the following:

```
deleteEvenNumbers( pHead);
displayList( pHead);
```
and it would result in the following output:
```
3->5->9->11
```

4. Insert listB into the middle of listA, after the list value n. The function header would be:
```
insertList( Node *pListA, Node *pListB, int insertValue)
```
Assume list A and list B contain the following:
pListA is: 2->3->9->11
pList B is: 5->6->8
Calling `insertList(…)` from within main would look like the following:
```
cout << "Enter value for insertion: ";
cin >> userInput;
insertList( pListA, pListB, userInput);
displayList( pListA);
```
and it would result in the following output:
```
Enter value for insertion: 3
2->3->5->6->8->9->11
```

5. Sum lists: Given list A and list B of the same length, traverse them to create a new list, where the first new list node contains the sum of the first node on A plus the first node on B, the second new list node contains the sum of the second node on A plus the second node on B, and so on.
Assume list A and list B contain the following:
pListA is: 1->3->4->7
pListB is: 2->4->6->10
Calling `combineLists(…)` from within main would look like the following:
```
combineLists( Node *&pHead, Node *pListA, Node *pListB);
displayList( pHead);
```
and it would result in the following output:
```
3->7->10->17
```