Lab 5 - Exercises[i]

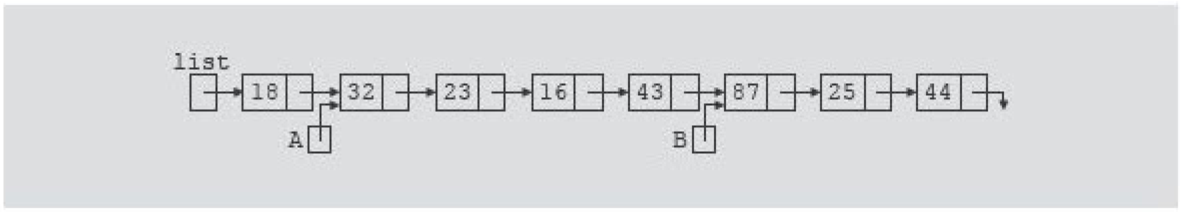


Figure 5-1 Linked List for Exercises 2 & 4

Chapter 5-2.What is the output of each of the following C++ statements?

```
a. cout << list->info;
b. cout << A->info;
c. cout << B->link->info;
d. cout << list->link->link->info
```

Chapter 5-4. Mark each of the following statements as valid or invalid. If a statement is invalid, explain why.

```
a. A = B;
b. list->link = A->link;
c. list->link->info = 45;
d. *list = B;
e. *A = *B;
f. B = A->link->info;
g. A->info = B->info;
h. list = B->link->link;
i. B = B->link->link->link;
```

Chapter 5-14. What is the output of the following program segment?

```
list<int> intList;
ostream_iterator<int> screen(cout, " ");
list<int>::iterator listIt;
intList.push_back(5);
intList.push_front(23);
intList.push_front(45);
intList.pop_back();
intList.push_back(35);
intList.push_front(0);
intList.push_back(50);
intList.push_front(34);
copy(intList.begin(), intList.end(), screen);
cout << endl;
listIt = intList.begin();
intList.insert(listIt,76);
++listIt;
++listIt;
intList.insert(listIt,38);
intList.pop_back();
++listIt;
++listIt;
intList.erase(listIt);
intList.push_front(2 * intList.back());
intList.push_back(3 * intList.front());
copy(intList.begin(), intList.end(), screen);
cout << endl;
```

Lab 5 - Programming Exercises[i]

Chapter 5-6.

a. Add the following operation to the `class orderedLinkedList`:

```
void mergeLists(orderedLinkedList<Type> &list1,
                orderedLinkedList<Type>  &list2);
   // This function creates a new list by merging the
   // elements of list1 and list2.
   // Postcondition: first points to the merged list; list1
   // and list2 are empty
```

Example: Consider the following statements:

```
orderedLinkedList<int> newList;
orderedLinkedList<int> list1;
orderedLinkedList<int> list2;
```

Suppose `list1` points to the list with the elements `2 6 7` and `list2` points to the list with the elements `3 5 8`. The statement: `newList.mergeLists(list1, list2)`; creates a new linked list with the elements in the order `2 3 5 6 7 8` and the object `newList` points to this list. Also, after the preceding statement executes, `list1` and `list2` are empty.

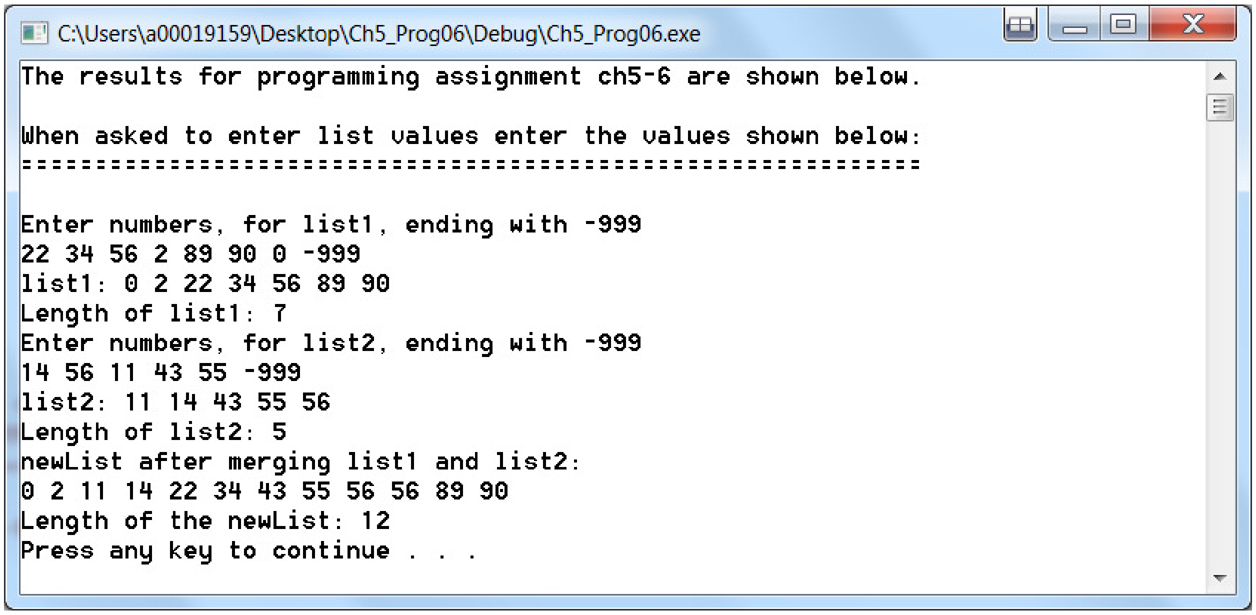
Implementation Notes: No additional memory is required when creating the merged list. The correct solution should perform the following tasks:

1. Create a pointer to the merged list (`newList`)
2. Compare the first element in `list1` and `list2`
3. Insert the smaller of the 2 elements into the merged list through pointer manipulation.
4. Advance the "`first`" pointer of the list containing the smaller element.
   a. The size of the list containing the smaller element is reduced by 1.
5. Repeat the process until one of the lists is empty.
6. The elements in the remaining list should then be added to the end of the merged list.
7. Don't forget to update the "`count`" variable in each list after removing an element from that list.
8. After the elements in both lists have been added to the merged list simply set the "`first`" and "`last`" pointers in each list to `NULL` to indicate the lists are empty. This does not result in a memory leak because the elements in each list were added to the merged list through pointer manipulation.

Modify the file `orderedLinkedList.h` and add your changes at the end of the `orderedLinkedList` class definition. The file `linkedList.h` required by the file `orderedLinkedList.h` has been provided as well.

b. Write the definition of the function template `mergeLists` to implement the operation `mergeLists`. Modify the class `orderedLinkedList` in the file `orderedLinkedList.h` and add your function definition at the end of the file. Then use the test program lab5-6.cpp to test your program. Use the input data (if any) shown in the output window on the following pages and then compare your results with the expected results.

Expected Program Input and Output



```
The results for programming assignment ch5-6 are shown below.

When asked to enter list values enter the values shown below:
=============================================================

Enter numbers, for list1, ending with -999
22 34 56 2 89 90 0 -999
list1: 0 2 22 34 56 89 90
Length of list1: 7
Enter numbers, for list2, ending with -999
14 56 11 43 55 -999
list2: 11 14 43 55 56
Length of list2: 5
newList after merging list1 and list2:
0 2 11 14 22 34 43 55 56 56 89 90
Length of the newList: 12
Press any key to continue . . .
```

[i] Exercises and Programming Exercises were extracted from the adopted course textbook, Data Structures Using C++, Second Edition, D.S. Malik, 2010 Course Technology, Cengage Learning for the purpose of annotating each exercise with Professor specific instructions (when necessary) that facilitate student completion.