PSet listdbl: a doubly-linked list

Table of Contents

Step 1: sorted()....1
Step 2: push_sorted()*....2
Step 3: swap_pairs()....2
Submitting your solution....3
Files to submit....3
Due and Grade points....3

- This Pset is an extension of Lab8.
  It covers the Advanced Operations of Doubly Linked Lists.
- You continue on coding by using listdbl.cpp that you coded in Lab8.
  - listdbl.h – Don't change this file.
  - driver.cpp – Don't change this file.
  - listdbl.cpp – Begin with the file you coded in Lab8 file.
  - listdblx.exe, listdblx – Provided for your references only. It may have bugs.

Step 1: sorted()

The two **sorted() functions** check whether or not the list is sorted in either ascending or descending order.

The following **sorted()** checks for the ascending order using ::less() first. Then if it successful, it returns **true**, otherwise it checks for the descending order again as shown below.

```
bool sorted(pList p) {
      return sorted(p, ::less) || sorted(p, more);
}
```

This following **sorted()** does the actual checking of sorting. If the size of the list is one or two are always sorted. For example, the function returns

- `true if p: 1 2 3, comp: less`
- `false if p: 1 2 3, comp: more`
- `true if p: 1 2 2 3 7, comp: less`
- `true if p: 7 7 2 1 1, comp: more`

```
bool sorted(pList p, bool (*comp)(int a, int b)) {
      if (size(p) <= 1) return true;
      cout << "your code here\n";
      return true;
}
```

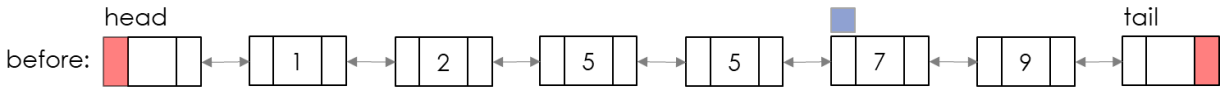
Step 2: push_sorted()*

The function **push_sorted()** used in option z inserts a new node with value in sorted order to the list in either ascending or descending order.

```
void push_sorted(pList p, int value)
```

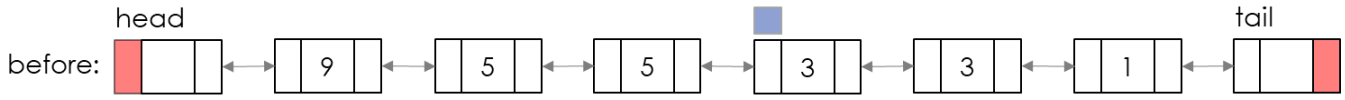
If the list is sorted in ascending, you may invoke **insert()** with the position node x of which the value is greater than **value**. Use **more()** to find the node x which is greater than value, and **less()** for descending case.

To insert **value = 5**, for example, you must find the position node x with **value = 7** using **more()** function.,



Take the similar steps for descending ordered list.

To insert **value = 5**, for example, you must find the position node x with **value = 3** using **less()** function.,



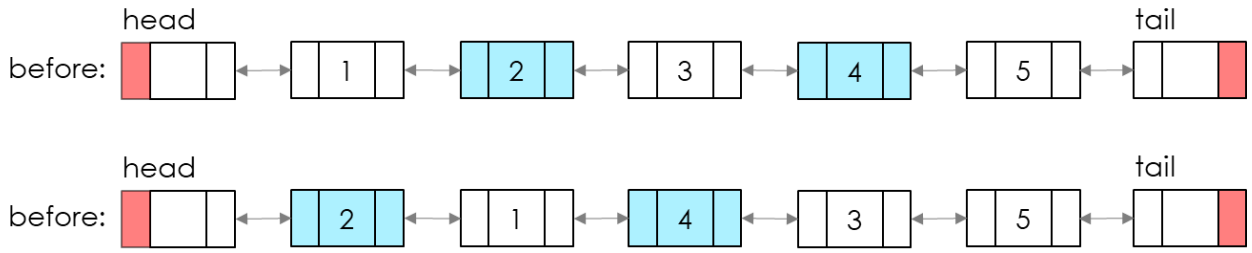
Hint: You may need the following functions to implement this part:
sorted(), insert(), more(), less()

```
// inserts a new node with value in sorted order
void push_sorted(pList p, int value) {
  if sorted(p, ::less)
     insert("...find a node more() than value", value);
  else if (...)
     insert("...find a node less() than value", value);
}
```

Step 3: swap_pairs()

This function is simple, but provide you with a good insight of a doubly linked list coding. It swaps two values of adjacent nodes in the list. It does not swap the links,

but the values only. If the list has an odd number of nodes, then the last one remains as it is. It goes through the list once, its time complexity is O(n).



Submitting your solution

- Include the following line at the top of your every file with your name signed. On my honour, I pledge that I have neither received nor provided improper assistance in the completion of this assignment. Signed: _____
- Make sure your code **compiles** and **runs** right before you submit it.
- If you only manage to work out the homework partially before the deadline, you still need to turn it in. However, don't turn it in if it does not compile and run.
- Place your source files in the folder you and I are sharing.
- After submitting, if you realize one of your programs is flawed, you may fix it and submit again as long as it is **before the deadline**. You may submit as often as you like. **Only the last version** you submit before the deadline will be graded.

Files to submit

- Submit the following files.
  - listdbl.cpp
- Use pset folder in piazza to upload your files.

Due and Grade points

- Due: 11:55 pm