Chp18- STL lists – Lab Problem

This lab aims to provide hands-on experience with the std::list container in C++, focusing on its core functionalities: insertion, deletion, searching, sorting, merging, splicing, and adding/removing elements from the ends.

Specifications:

Part 1: List Creation and Basic Operations

1. **Create Lists:**
    - Create two empty std::list<string> objects named lst1 and lst2.
    - Populate lst1 with the following string representations of integers using push_back(): "5", "12", "2", "8", "12".
    - Populate lst2 with the following string representations of integers using push_back(): "3", "9", "1", "7".
2. **Display Lists:**
    - Write a **template** function printList(const std::list<string>& lst) that iterates through the list and prints each string element separated by a space (use an iterator based navigation).
    - Use this function to display the initial contents of lst1 and lst2.
3. **Push and Pop:**
    - Use push_front() to add the string "15" to the beginning of lst1.
    - Use pop_back() to remove the last element from lst2.
    - Display the updated contents of both lists.

Part 2: Insertion and Erasure

1. **Insertion:**
    - Find the first occurrence of the string "12" in lst1 using std::find().
    - If found, insert the string "10" into lst1 **before** the found "12" using insert().
    - If not found, print a message indicating that "12" was not in the list.
    - Display the updated lst1.
2. **Erasure (Single Element):**
    - Find the first occurrence of the string "2" in lst1 using std::find().
    - If found, erase this element from lst1 using erase().
    - Display the updated lst1.
3. **Erasure (Range):**
    - Find the first occurrence of the string "8" in lst1.
    - Find the first occurrence of the string "15" in lst1.
    - If both are found and the iterator to "8" comes before the iterator to "15", erase the range of elements (inclusive of "8", exclusive of "15") from lst1 using erase() with a range of iterators.
    - Display the final lst1.

Part 3: Searching and Sorting

1. **Searching:**
    - Write a function findStringNumber(const std::list<string>& lst, const string& target) that uses std::find() to search for the target string in the list and returns true if found, false otherwise.
    - Use this function to search for "10" and "4" in lst1 and print whether each was found.
2. **Sorting (Lexicographical):**
    - Sort lst1 using its default sort() method. Observe the output. Why is the sorting order the way it is? (Hint: String comparison is *lexicographical*).
    - Display the sorted lst1.
3. **Sorting (Numerical):**
    - Write a custom comparison function (or lambda) that compares two strings by converting them to integers using std::stoi() and then comparing the integer values.
    - Sort lst1 again using the custom comparison function with the sort() method.
    - Display the numerically sorted lst1.

Part 4: Merging and Splicing

1. **Merging:**
    - Ensure lst1 and lst2 are numerically sorted (use the custom comparison from Part 3 if needed).
    - Merge the contents of lst2 into lst1 using the merge() method. Provide the custom comparison function to ensure numerical merging.
    - Display the merged lst1. Note that lst2 will be empty after the merge.
2. **Splicing:**
    - Create a new std::list<string> called tempList and populate it with "99", "88", "77".
    - Find the element "9" in the merged lst1.
    - Splice the entire tempList into lst1 at the position before "9" using the splice() method.
    - Display the updated lst1 and the (now empty) tempList.
3. **Splicing (Range):**
    - Create another std::list<string> called anotherList with "16", "18", "22", "24".
    - Find the iterators to the second and last elements of anotherList.
    - Splice the range of elements (from the second to the second-to-last) from anotherList into lst1 at the beginning.
    - Display the final lst1 and the modified anotherList.

Part 5: Removing Duplicates

1. **Removing Consecutive Duplicates:**
    - Add a duplicate value (e.g., "12") to lst1 using push_back().
    - Sort lst1 numerically.
    - Use the unique() method to remove consecutive duplicate elements.
    - Display the lst1 after removing duplicates.

Sample Output

```
Part 1: Creation and Basic Operations --------------------------------
Initial lst1: 5 12 2 8 12
Initial lst2: 3 9 1 7
lst1 after push_front("15"): 15 5 12 2 8 12
lst2 after pop_back(): 3 9 1

Part 2: Insertion and Erasure ---------------------------------
lst1 after inserting "10" before "12": 15 5 10 12 2 8 12
lst1 after erasing "2": 15 5 10 12 8 12

Part 3: Searching and Sorting ---------------------------------
Found "10" in lst1? YES
Found "4" in lst1? NO
lst1 after lexicographical sort: 10 12 12 15 5 8
lst1 after numerical sort: 5 8 10 12 12 15

Part 4: Merging and Splicing ---------------------------------
lst2 after numerical sort (before merge): 1 3 9
lst1 after merging lst2: 1 3 5 8 9 10 12 12 15
lst2 after merging:
lst1 after splicing tempList before "9": 1 3 5 8 99 88 77 9 10 12 12 15
tempList after splicing:
lst1 after splicing range from anotherList: 18 22 24 1 3 5 8 99 88 77 9 10 12 12 15
anotherList after splicing range: 16

Part 5: Removing Duplicates ---------------------------------
lst1 before removing duplicates: 1 3 5 8 9 10 12 12 12 15 18 22 24 77 88 99
lst1 after removing consecutive duplicates: 1 3 5 8 9 10 12 15 18 22 24 77 88 99

All done!
```