COMP 2611 – Data Structures

Lab #2 (September 18-20, 2023)

Instructions

Download `Lab2-Files.zip` and unzip the archive. You will obtain three sub-folders, each of which contains a Dev-C++ project, `LinkedList.dev`. For each problem, open the project and modify the files as described below.

1. Folder: LinkedList-NonRecursive

The file `LinkedList.h` contains prototypes for the functions listed in Table 1:

Function	Description
`Node * predecessor (Node * top, int n);`	Suppose *n* was inserted in the linked list. This function returns the node that would come immediately before *n* in the linked list. If no such node exists, it returns NULL.
`Node * insertSorted (Node * top, int n);`	This function inserts a node with the value *n* in the linked list and returns the top of the list after the node is inserted. It must use the *predecessor* function.
`bool isEqual (Node * top1, Node * top2);`	Returns *true* if the two linked lists contain the same elements in the same order and *false*, otherwise.
`Node * copyList (Node * top);`	Makes a duplicate of the linked list passed as a parameter and returns the top of the new list.
`void clear (Node * top);`	Deletes all the elements from the linked list passed as a parameter.
`Node * merge (Node * top1, Node * top2);`	Merges the two linked lists passed as parameters, assuming that they are in sorted order. Returns the top of the merged list. No new nodes should be created.

Table 1: List of Non-Recursive Linked List Functions

(a) The code for the *predecessor* function is provided in `LinkedList.cpp`. Write the code for the other functions listed in Table 1 in `LinkedList.cpp`.

(b) If you need assistance in writing the *merge* function, you can look at the code in `Merge.cpp` (included in the folder but not part of the project). You may copy all the code in `Merge.cpp` to the relevant section of `LinkedList.cpp`.

(c) Code has already been written in `UsingLinkedList.cpp` to test the functions listed in Table 1. Compile and test the program. Ensure that the correct results are obtained.

2. Folder: LinkedList-Recursive

The file `LinkedList.h` contains prototypes for the recursive functions listed in Table 2:

Function	Description
`void printListRec (Node * top);`	Displays the elements of the linked list.
`void printListReverseRec (Node * top);`	Displays the elements of the linked list in reverse order.
`int sizeRec (Node * top);`	Returns the number of elements in the linked list.
`bool containsRec (Node * top, int key);`	Returns *true* if the linked list contains *key* and *false*, otherwise.
`int sumRec (Node * top);`	Returns the sum of the elements in the linked list.

Table 2: List of Recursive Linked List Functions

(a) Write the code for the functions listed in Table 2 in `LinkedList.cpp`.
(b) In the *main* function of `UsingLinkedList.cpp`, insert 17, 23, and 250, and 45 at the tail of a list and print the list using the *printListRec* function. Using *sizeRec*, find the number of elements in the list. Using *sumRec*, find the sum of the elements in the linked list. Finally, using the *containsRec* function, determine if the linked list contains a certain value input by the user.
(c) Compile and test the program. Ensure that the correct results are obtained.

3. Folder: LinkedList-Recursive2

The file `LinkedList.h` contains prototypes for the recursive functions listed in Table 3:

Function	Description
`bool isEqualRec (Node * top1, Node * top2);`	Returns *true* if the two linked lists contain the same elements in the same order and *false*, otherwise.
`Node * copyListRec (Node * top);`	Makes a physical copy of the linked list passed as a parameter and returns the top of the new list.
`Node * insertSortedRec (Node * top, int n);`	Inserts the value *n* in the linked list assuming that the list is maintained in ascending order. Returns the top of the list after *n* is inserted.
`Node * mergeRec (Node * top1, Node * top2);`	Merges the two linked lists passed as parameters, assuming that each list is maintained in ascending order. Returns the top of the merged list. No new nodes should be created.

Table 3: List of Additional Recursive Linked List Functions

(a) Write the code for the functions listed in Table 3 in `LinkedList.cpp`.
(b) Code has already been written in `UsingLinkedList.cpp` to test the functions in Table 3. Compile and test the program. Ensure that the correct results are obtained.

END OF LAB #2