**Assignment 4 report**

By: Malak Mekky

ID: 900222132

**Output:**

Creating the list:

Text

Description automatically generated

Adding to list:

Text

Description automatically generated

Sum of nodes:



Removal of node, with new linked list and sum outputted:

Graphical user interface, text, application

Description automatically generated

Screenshot of full terminal:

Text

Description automatically generated

The program outputs the problem correctly.

The LinkedList class contains member functions to add and remove nodes, print the contents of the list, create a list from a vector of integers, and calculate the sum of all the values. The code defines a Node struct that contains an integer value, an integer count that represents the number of times the value occurs in the list, and a pointer to the next node in the list.

It creates a linked list using the data structure vector, The function createList takes a vector<int> as a parameter and iterates through its elements to add them to the linked list. The program asks the user to enter the values to be added to the linked list to create the list. It also asks

**Explanations:**

*LinkedList.h*

It first defines a Node struct that contains 2 variables of type int: value and count. And a pointer called next, which points to the next node in the list.

The LinkedList class defines public member functions: add, remove, printList and createList. The functions adds nodes, removes nodes, outputs the contents of the linked list and creates a linked list respectively. There is also a private member pointer which points at the head of the list.

Outside of the class, there is a function definition called insert, which takes a reference to the vector of integers (the linked list), an integer which will be searched for in the linked list, and the integer which is to be inserted in the linked list.

*Create list function:*

The function initializes the linked list using the values from the input vector.

The function takes a *vector<int>* as a constant reference, which contains the values to be added to the linked list. The function iterates over each element in the input vector and adds it to the linked list using the *add* function.

*Add and insert function:*

The *add* function from the *LinkedList* class adds a new node to the end of the linked list, whereas the *insert* function adds it to wherever the user would like to insert the node.

Add function: If the linked list is empty, the function would create a new node and assign it as the head of the list (make the head pointer head to this node). But if there are already values in the list, the function traverses the list and looks for if the value already exists there. If the value already exists, it increments the count of that node. But if the value does not exist, the function traverses to the end of the list till it reaches the end then appends to the list.

Insert function: The function calls a reference to the linked list *(vector<int>& values),* an integer which will be searched for *(num1)*, and the integer which is to be inserted *(num2).* The function iterates through each element of the list, checking if the current element has the same value as *num1*. If an element matches, the function will use the insert method from the vector class to insert *num2* immediately after the current element. Then increments counter by 1 to skip over the newly inserted element.

*Remove function:*

The boolean function *remove* calls an integer ‘*value’* as a parameter. It looks for the first node in the linked list which has a matching value. If the node is found and removed, it will return *true*, otherwise, it will return *false*.

It begins by creating two pointers of type *Node*\*: ‘*current’* and ‘*prev’*, which are initialized to the *head* of the list and *nullptr* respectively. The function then traverses through the linked list, and checks the value of *current* against *value*. If the value is found, there are two ways of removing the node:

* If the node has a count larger than 1, it decrements the count by one.
* If the node has a count of 1, the function removes the node from the linked list.
  + If the node was the first value (aka the *head*), the value of *head* will change to point to the next node. And returns *true.*
  + Otherwise, *prev* is updated to point to the node after the removed node, and the function returns *true.*

If the function reaches the end of the linked list without finding a node with a matching value, it returns *false*.

*sumNodes function:*

The *sumNodes* function calculates the sum of all nodes in the linked list by multiplying the value of each node by its count, and then adding up all the results.

The function initializes a variable *sum*, and then iterates over each node in the linked list. For each node, it multiplies the node's value by its count and adds the result to *sum*. The function then moves to the next node, repeating the process until it reaches the end of the linked list.

*printList function:*

The printList function outputs the values of all nodes in the linked list along with their corresponding count. The function initializes a pointer variable *current* to the first node of the linked list, and then iterates over each node in the list using a loop. For each node, it prints the node's value and count. The function then moves to the next node by updating the *current* pointer to the next node, repeating the process until it reaches the end of the linked list.