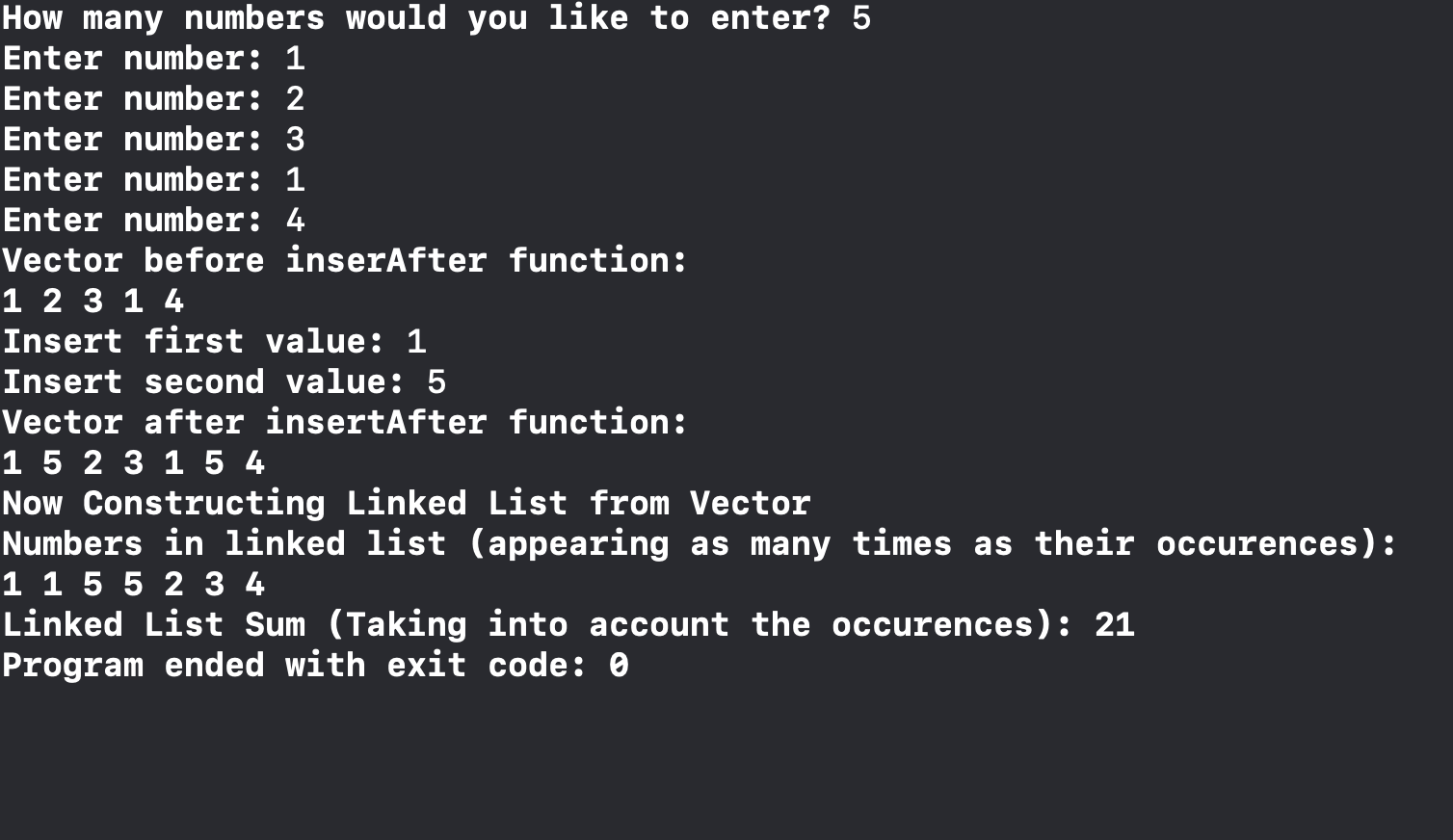
Mohamed Azouz 900225230

**CS2 Assignment 4 Report**

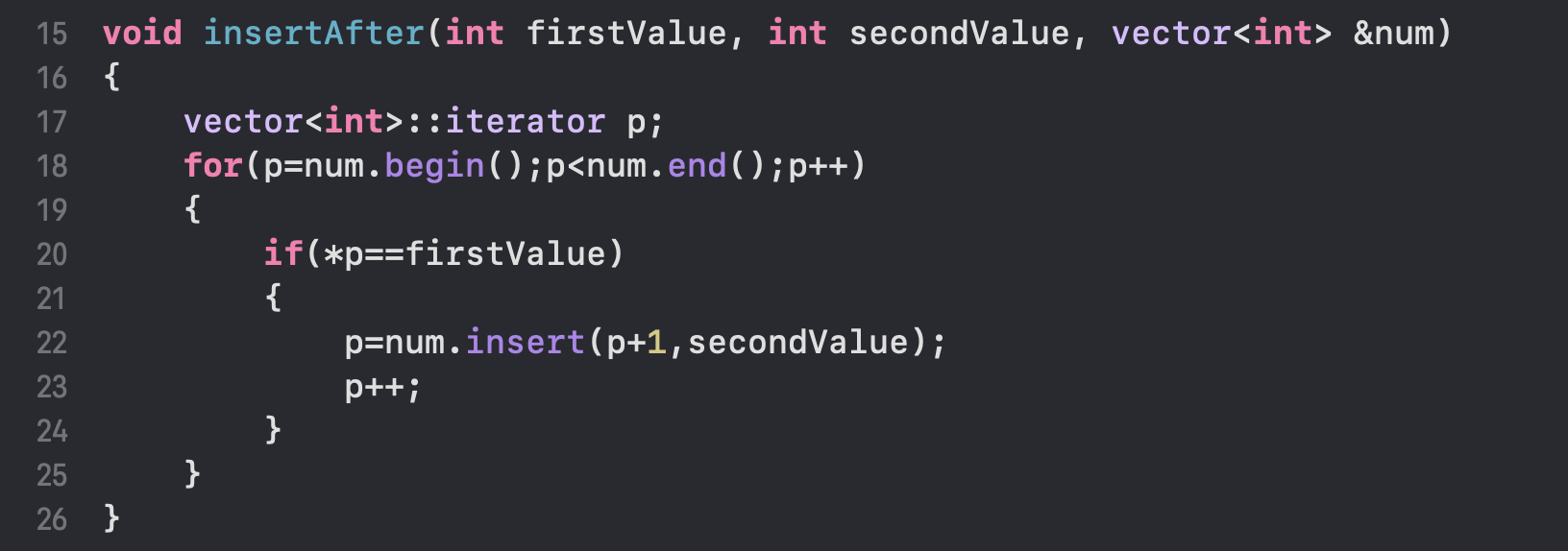
**Code Output:**

****

* **insertAfter(int firstValue, int secondValue, vector<int> &num)**

The insertAfter function, as described in the assignment description, is a function which inserts a certain number after every occurrence of another number in a vector of integer elements.

This is my code for the insertAfter function:



The function accepts 3 parameters:

* 2 integers ‘firstValue’ and ‘secondValue’
* 1 vector of type int

The vector is passed by reference to ensure that any changes made (numbers inserted) are reflected in the main function.

The function uses an iterator ‘p’ to iterate through the vector. An iterator is necessary as it is not a regular array, and it is preferable to use an iterator when dealing with vectors. The function compares all values in the vector to the ‘firstValue’ variable. If they are equal, the ‘secondValue’ is inserted in the ‘p+1’ position, meaning right after. ‘p’ is also incremented after a number is inserted. This is necessary because if both values were equal, then it would cause an infinite loop as it will check the value it just inserted.

* **class LinkedLists**

The ‘LinkedLists’ class contains 3 main parts besides the functions:

* struct Node

The node struct contains three variables. There are the ‘num’ and ‘occurrences’ variables which are of type int and contain the number inputted by the user (in this case from the vector). There is also a pointer node ‘next’ which points to the next node in the linked list.

* Node \* head

The node ‘head’ points to the first node in the linked list. It is initialized as NULL in the constructor as it creates an empty list at first.

* Node \* temp

The node ‘temp’ points to the last node in the linked list. It is used for quicker access when inserting a node to the end of the list instead of using a while loop to iterate through every time.

Then there are the functions of the class:

* LinkedLists()

The ‘LinkedLists’ constructor initializes the object to contain an empty linked lists. It does this by setting the head = NULL and the temp = head.

* void add(int n)

The add function takes an integer and adds it to the linked list. Normally, any integer received would create a new node. However, to accommodate this assignment, the function searches the current linked list for the inserted number. If the number is present, it increments its occurrences to reflect the number of times it appears. If it is not found, a new node is created and the occurrences are initialized with 1.

* void remove(int index)

The remove function takes an index and removes the node in the linked list at that index. This could have also been done using the number which would have altered the code slightly. When a node is being deleted, its previous node is made to point at the node after it, then it can be deleted.

* void print()

The print function displays all the numbers in the linked list. I programmed it so that it will display each number the amount of times it occurred in the vector.

* void constructFromVector(vector<int> v)

This function accepts a vector made of ints and inserts them into the linked list. This is very simple as it just iterates through the vector and uses the add function on all the numbers.

* int sum()

This function returns the sum of all the elements in the linked list, taking into account their number of occurences. It does this by adding up each element in the list multiplied by its occurrences.

* ~LinkedLists()

This destructor is needed to free up the memory as there are a lot of pointers used in the Linked List.

**Reference for Linked Lists:**

* **https://www.geeksforgeeks.org/what-is-linked-list/**