Lab task 8

#include <iostream>

#include <string>

Using namespace std;

// Define the structure for a student node

Struct Student {

String name;

String sap\_id;

Student\* next; // Pointer to the next node

};

// Function to insert a new student at the end of the linked list

Void insertStudent(Student\*& head, const string& name, const string& sap\_id) {

Student\* newStudent = new Student();

newStudent->name = name;

newStudent->sap\_id = sap\_id;

newStudent->next = nullptr;

if (head == nullptr) {

head = newStudent; // If list is empty, make the new student the head

} else {

Student\* temp = head;

While (temp->next != nullptr) {

Temp = temp->next; // Traverse to the end of the list

}

Temp->next = newStudent; // Link the new student

}

}

// Function to delete a student at a specific position

Void deleteStudent(Student\*& head, int position) {

If (head == nullptr) {

Cout << “List is empty. Cannot delete.” << endl;

Return;

}

Student\* temp = head;

// If head needs to be removed

If (position == 1) {

Head = temp->next; // Change head

Delete temp; // Free old head

Return;

}

// Find the previous node of the node to be deleted

For (int I = 1; temp != nullptr && I < position – 1; i++) {

Temp = temp->next;

}

// If position is more than the number of nodes

If (temp == nullptr || temp->next == nullptr) {

Cout << “Position “ << position << “ does not exist.” << endl;

Return;

}

// Node temp->next is the node to be deleted

Student\* next = temp->next->next; // Store pointer to the next node

Delete temp->next; // Free memory

Temp->next = next; // Unlink the deleted node from the list

}

// Function to display the linked list

Void displayList(Student\* head) {

If (head == nullptr) {

Cout << “The list is empty.” << endl;

Return;

}

Student\* temp = head;

Int count = 1;

While (temp != nullptr) {

Cout << “Student “ << count << “: Name = “ << temp->name << “, SAP ID = “ << temp->sap\_id << endl;

Temp = temp->next;

Count++;

}

}

Int main() {

Student\* head = nullptr; // Initialize head of the linked list

Int numStudents;

// Input number of students

Cout << “Enter the number of students (at least 5): “;

Cin >> numStudents;

// Ensure that at least 5 students are entered

If (numStudents < 5) {

Cout << “Please enter at least 5 students.” << endl;

Return 1;

}

// Input student details

For (int I = 0; I < numStudents; i++) {

String name, sap\_id;

Cout << “Enter name of student “ << (I + 1) << “: “;

Cin >> name;

Cout << “Enter SAP ID of student “ << (I + 1) << “: “;

Cin >> sap\_id;

insertStudent(head, name, sap\_id); // Insert student into the linked list

}

// Display the linked list before deletion

Cout << “\nStudent records before deletion:\n”;

displayList(head);

// Delete 2nd and 5th student

deleteStudent(head, 2);

deleteStudent(head, 4); // After deleting the 2nd student, the 5th student becomes the 4th

// Display the linked list after deletion

Cout << “\nStudent records after deletion:\n”;

displayList(head);

// Free remaining memory

While (head != nullptr) {

Student\* temp = head;

Head = head->next;

Delete temp;

}

Return 0;

}