Stella or margie

// Function to add a student to the linked list

void addStudent(Node\*& head, const Student& student) {

Node\* newNode = new Node{ student, nullptr };

if (!head) {

head = newNode;

} else {

Node\* temp = head;

while (temp->next) {

temp = temp->next;

}

temp->next = newNode;

}

}

dito natin ma dadagdag ang mga student gamit ang function na ito

**Node\* newNode = new Node{ student, nullptr };** = meaning gumagawa tayo nito ng bagong node, may dalawang part ang node Student at Nullptr

**Student** = basta information ng student yung mga nilalagay nyo yung greds etcc

**Nullptr** = ito naman ay part sya ng linked list na ang ibigsabihin ay para malaman kung ang isang node ay ang huling node, at pinapakita nito na walang tinutukoy na value ang pointer

**if (!head) { head = newNode; }** = Kung walang unang node sa linked list (ibig sabihin, ang head ay nullptr), ilalagay natin ang bagong newNode bilang unang node sa listahan. Kapag wala pang estudyante sa listahan, ang head ay magiging newNode.

**else { Node\* temp = head; while (temp->next) { temp = temp->next; } temp->next = newNode;** } = kung may nag eexist nang node sa student meaning ang nullptr ay hindi na ang head magsisimula na ito sa new node

**while (temp->next)** = ito naman ang tumutukoy para mag tuloy tuloy ang mga node parang train

Kapag naabot natin ang huling node (kung saan ang next ay nullptr), ilalagay natin ang newNode bilang susunod na node ng kasalukuyang huling node (temp->next = newNode)

Pag nag tuloy na sa add student dito na sya mag pproceed

**Si jayr dito ah wag nyo intindihin to hAHAHAHAHHA sinama ko lang (itong baba ah)**

**void createStudent(Node\*& head) {**

**Student newStudent;**

**cout << "Enter Student ID: ";** = dito mo ilalagay ang ID

**cin >> newStudent.id;** = so bali mapupunta na sya sa

**cin.ignore(); // Ignore leftover newline** = meaning naman nito ay iignorin nya lang yung pag enter lang lalagay mo since kadalas natin na mimiss click sa enter minsan napipindot natin to ng double

**cout << "Enter Student Name: ";**

**getline(cin, newStudent.name);**

**cout << "Enter Quiz 1 Score (out of 50): ";**

**cin >> newStudent.quiz;**

**cin.ignore(); // Ignore leftover newline**

**cout << "Enter Quiz 2 Score (out of 50): ";**

**cin >> newStudent.quiz2;**

**cin.ignore(); // Ignore leftover newline**

**cout << "Enter Recitation Score (out of 100): ";**

**cin >> newStudent.recitation;**

**cin.ignore(); // Ignore leftover newline**

**cout << "Enter Case Study Score (out of 100): ";**

**cin >> newStudent.caseStudy;**

**cin.ignore(); // Ignore leftover newline**

sa calculate grade naman

tatawagin nya ang function na newStudent.grade and yung naka = nayun doon mapupunta ang mga value na nakuha natin sa cin sa student input

**// Calculate grade**

**newStudent.grade = calculateGrade(newStudent.quiz, newStudent.quiz2, newStudent.recitation, newStudent.caseStudy);**

after makuha dito sya mapupunta at I sosolve nya ito (explain nalang to since Madali nalang)

**// Function to calculate the grade in percentage, capped at 99**

**double calculateGrade(double quiz, double quiz2, double recitation, double caseStudy) {**

**double totalScore = ((quiz / 50.0) \* 30) + ((quiz2 / 50.0) \* 30) +**

**((recitation / 100.0) \* 20) + ((caseStudy / 100.0) \* 20);**

**return (totalScore > 99) ? 99 : totalScore;**

**}**

After makuha ang mga sagot or ma solve ang sagot

Icocompare nya ang total average sa setStudentStatus

**setStudentStatus(newStudent); // Set status based on Philippine grading standard**

**// Function to set student's status based on Philippine grading standards**

**void setStudentStatus(Student& student) {**

**if (student.grade >= 90) {**

**student.status = "Excellent";**

**} else if (student.grade >= 80) {**

**student.status = "Very Good";**

**} else if (student.grade >= 75) {**

**student.status = "Passed";**

**} else {**

**student.status = "Failed";**

**}**

**}**

Ayan pag ang average ng student ay nagging 90 above excellent…… dagdag nalang…..

After naman na nakuha ang student status mapupunta na sya sa linked list natin… kaya pwede na natin I view ang nagging result…

**addStudent(head, newStudent); // Add to linked list**

**cout << "Student added successfully!\n";** = after nyan mag sasabi sya ng ganito

**}**

Stella or margie

Node\* findStudentById(Node\* head, int id) {

Node\* temp = head;

while (temp) {

if (temp->student.id == id) {

return temp;

}

temp = temp->next;

}

return nullptr;

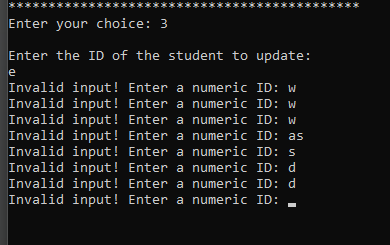
}

dito naman natin ma sesearch ang student gamit ang function na ito

**Node\* temp = head;** //= Ginagawa natin ang isang temporary pointer or (temp) na magsisimula sa unang node ng listahan, at Ang (head) ay ang unang node ng linked list, kaya magsisimula tayo dito para hanapin ang student.

**while (temp**) //= Ang while loop ay nagpapatuloy hangga't may tinutukoy pa ang temp. Kung ang temp ay nagging nullptr, ibig sabihin natapos na ang linked list at wala nang node na matutukoy, kaya ang loop ay titigil.

Eto example…



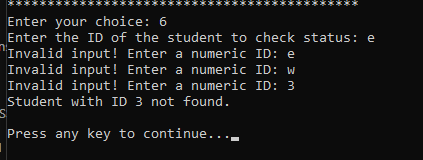
Hanggat di mo sinasabi ang number since ang id ay number sya mag loloop sya

**if (temp->student.id == id)** = Tinitingnan nito kung ang id ng kasalukuyang node (temp->student.id) ay tumutugma sa id na hinahanap. Kungtugma naman ang ID, ibig sabihin nahanap na ang estudyante, kaya ibabalik ang kasalukuyang node **(return temp).**

**temp = temp->next; =** Kung hindi pa natutugma ang id, lilipat tayo sa susunod na node. Ginagamit ang temp->next upang magpatuloy sa susunod na node sa linked list. Ibinabalik nito ang temp sa susunod na node, at ipagpapatuloy ang paghahanap sa susunod na node.

**return nullptr; =** Kung matapos mag-loop sa lahat ng nodes ay hindi nahanap ang estudyante (ibig sabihin walang node na may matching id), magbabalik ng nullptr meaning hindi nahanap ang estudyante sa listahan.

Eto example



Pag hindi nag match sa id or hindi nag eexist ang id na hinahanap mag rereturn na sya sa simula….

Stella or margie

void deleteStudent(Node\*& head, int id) {

if (!head) return;

if (head->student.id == id) {

Node\* temp = head;

head = head->next;

delete temp;

cout << "Student with ID " << id << " deleted successfully!\n";

return;

}

Node\* prev = head;

Node\* curr = head->next;

while (curr) {

if (curr->student.id == id) {

prev->next = curr->next;

delete curr;

cout << "Student with ID " << id << " deleted successfully!\n";

return;

}

prev = curr;

curr = curr->next;

}

cout << "Student with ID " << id << " not found.\n";

}

Dito naman natin ma dedelete ang student sa function na ito

**if (!head) return;** = If head is nullptr, the function immediately exits, meaning the list is empty, so there’s nothing to delete.

Check if the Student to Delete is at the Head:

Kung ang student ID na sinesearch natin ay nag match sa head node ito ay magiging

**if (head->student.id == id) {**

**Node\* temp = head; =**  Store the **head** node in a temporary pointer **temp**.

**head = head->next; =**  Move the **head** pointer to the next node (**head = head->next**), effectively removing the head node from the list.

**delete temp; =**  Delete the old **head** node using delete **temp**, freeing up memory.

**cout << "Student with ID " << id << " deleted successfully!\n";** = tas ipprint na ang message na deleted successfully

**return;**

**}**

**Node\* prev = head;** = is a pointer that starts at the head node. It will keep track of the node just before the one we want to delete.

**Node\* curr = head->next;** = is a pointer that starts at the second node (head->next). It will move through each node in the list, checking for a matching ID.

// sa madaling salita yung dalawa nato function nya is pag hindi nag match sa unang ID ang sinerch ang head ay mag nenext hanggang sa mag match yung sineserch na ID

//pag yung loop ay nahanap na or nayari na sa dulo

**while (curr) {**

**if (curr->student.id == id) {** = If curr->student.id matches the given id

**prev->next = curr->next;** = Adjusts prev->next to skip over curr, effectively removing it from the list.

**delete curr;** = Deletes curr to free memory.

**cout << "Student with ID " << id << " deleted successfully!\n";** = then print uli ng deleted successfully

return;

}

**If the id doesn’t match, the pointers move forward to the next nodes (prev = curr and curr = curr->next).**

prev = curr;

curr = curr->next;

}

**cout << "Student with ID " << id << " not found.\n";** = Pag talaga walang nahanap or nag match sa ID ito ay mag pprint ng error message na Student with ID (yung ininput ng user) not found

sakin na den to -jayr

// Function to update a student

void updateStudent(Node\*& head) {

int id;

cout << "\nEnter the ID of the student to update: ";

while (!(cin >> id)) {

cout << "Invalid input! Enter a numeric ID: ";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

Node\* studentNode = findStudentById(head, id);

if (studentNode) {

cout << "Enter new Quiz 1 score: ";

cin >> studentNode->student.quiz;

cout << "Enter new Quiz 2 score: ";

cin >> studentNode->student.quiz2;

cout << "Enter new Recitation score: ";

cin >> studentNode->student.recitation;

cout << "Enter new Case Study score: ";

cin >> studentNode->student.caseStudy;

cin.ignore(); // Ignore leftover newline

// Recalculate grade

studentNode->student.grade = calculateGrade(studentNode->student.quiz, studentNode->student.quiz2, studentNode->student.recitation, studentNode->student.caseStudy);

// Update status

setStudentStatus(studentNode->student);

cout << "Student details updated successfully!\n";

} else {

cout << "Student with ID " << id << " not found.\n";

}

}

// Function to edit a student

void editStudent(Node\*& head) {

int id;

cout << "\nEnter the ID of the student to edit: ";

while (!(cin >> id)) {

cout << "Invalid input! Enter a numeric ID: ";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

Node\* studentNode = findStudentById(head, id);

if (studentNode) {

cin.ignore(); // Ignore leftover newline

cout << "Enter new name for student with ID " << id << ": ";

getline(cin, studentNode->student.name);

cout << "Student name updated successfully!\n";

} else {

cout << "Student with ID " << id << " not found.\n";

}

}

void editStudent(Node\*& head) {

int id;

cout << "\nEnter the ID of the student to edit: ";

while (!(cin >> id)) {

cout << "Invalid input! Enter a numeric ID: ";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

Node\* studentNode = findStudentById(head, id);

if (studentNode) {

cin.ignore(); // Ignore leftover newline

cout << "Enter new name for student with ID " << id << ": ";

getline(cin, studentNode->student.name);

cout << "Student name updated successfully!\n";

} else {

cout << "Student with ID " << id << " not found.\n";

}

}