Second year Computer Engineering class, set A of students like Vanilla Ice-cream and set B of students like butterscotch ice-cream. Write C++ program to store two sets using linked list. Compute and display: Write C/C++ program to store two sets using linked list, compute and display: a. Set of students who like either vanilla or butterscotch or both b. Set of students who like both vanilla and butterscotch c. Number of students who like neither vanilla nor butterscotch  
  
  
#include <iostream>

using namespace std;

// Node structure to represent each student

struct Node {

int studentId;

Node\* next;

};

// Function to insert a student into the linked list

void insert(Node\*& head, int studentId) {

Node\* newNode = new Node();

newNode->studentId = studentId;

newNode->next = head;

head = newNode;

}

// Function to display a set of students

void display(Node\* head) {

if (head == nullptr) {

cout << "No students in this set." << endl;

return;

}

Node\* temp = head;

while (temp != nullptr) {

cout << temp->studentId << " ";

temp = temp->next;

}

cout << endl;

}

// Function to check if a student is in a set

bool isPresent(Node\* head, int studentId) {

Node\* temp = head;

while (temp != nullptr) {

if (temp->studentId == studentId) {

return true;

}

temp = temp->next;

}

return false;

}

// Function to compute the union of two sets

Node\* unionSets(Node\* setA, Node\* setB) {

Node\* resultSet = nullptr;

Node\* temp = setA;

while (temp != nullptr) {

insert(resultSet, temp->studentId);

temp = temp->next;

}

temp = setB;

while (temp != nullptr) {

if (!isPresent(setA, temp->studentId)) {

insert(resultSet, temp->studentId);

}

temp = temp->next;

}

return resultSet;

}

// Function to compute the intersection of two sets

Node\* intersectionSets(Node\* setA, Node\* setB) {

Node\* resultSet = nullptr;

Node\* temp = setA;

while (temp != nullptr) {

if (isPresent(setB, temp->studentId)) {

insert(resultSet, temp->studentId);

}

temp = temp->next;

}

return resultSet;

}

// Function to compute the number of students who like neither vanilla nor butterscotch

int countNeither(Node\* setA, Node\* setB, int totalStudents) {

int count = 0;

for (int i = 1; i <= totalStudents; ++i) {

if (!isPresent(setA, i) && !isPresent(setB, i)) {

count++;

}

}

return count;

}

int main() {

Node\* setA = nullptr; // Set of students who like vanilla

Node\* setB = nullptr; // Set of students who like butterscotch

int numVanilla, numButterscotch, studentId;

// Get input for Set A (students who like vanilla)

cout << "Enter number of students who like vanilla ice-cream: ";

cin >> numVanilla;

cout << "Enter the student IDs for vanilla lovers: ";

for (int i = 0; i < numVanilla; ++i) {

cin >> studentId;

insert(setA, studentId);

}

// Get input for Set B (students who like butterscotch)

cout << "Enter number of students who like butterscotch ice-cream: ";

cin >> numButterscotch;

cout << "Enter the student IDs for butterscotch lovers: ";

for (int i = 0; i < numButterscotch; ++i) {

cin >> studentId;

insert(setB, studentId);

}

// Display sets

cout << "Students who like vanilla ice-cream (Set A): ";

display(setA);

cout << "Students who like butterscotch ice-cream (Set B): ";

display(setB);

// Union of Set A and Set B (Students who like either vanilla or butterscotch or both)

Node\* unionSet = unionSets(setA, setB);

cout << "Union of Set A and Set B (Students who like either vanilla or butterscotch or both): ";

display(unionSet);

// Intersection of Set A and Set B (Students who like both vanilla and butterscotch)

Node\* intersectionSet = intersectionSets(setA, setB);

cout << "Intersection of Set A and Set B (Students who like both vanilla and butterscotch): ";

display(intersectionSet);

// Number of students who like neither vanilla nor butterscotch

int totalStudents;

cout << "Enter total number of students: ";

cin >> totalStudents;

int neitherCount = countNeither(setA, setB, totalStudents);

cout << "Number of students who like neither vanilla nor butterscotch: " << neitherCount << endl;

return 0;

}