Assognment . 10

#include <iostream>

using namespace std;

struct Node {

int rollNo;

Node\* next;

};

class LinkedList {

public:

Node\* head;

LinkedList() {

head = nullptr;

}

void insert(int rollNo) {

Node\* newNode = new Node();

newNode->rollNo = rollNo;

newNode->next = head;

head = newNode;

}

bool search(int rollNo) {

Node\* temp = head;

while (temp != nullptr) {

if (temp->rollNo == rollNo) return true;

temp = temp->next;

}

return false;

}

void unionSets(LinkedList& setB, LinkedList& result) {

Node\* temp = head;

while (temp != nullptr) {

result.insert(temp->rollNo);

temp = temp->next;

}

temp = setB.head;

while (temp != nullptr) {

if (!search(temp->rollNo)) result.insert(temp->rollNo);

temp = temp->next;

}

}

void intersectionSets(LinkedList& setB, LinkedList& result) {

Node\* temp = head;

while (temp != nullptr) {

if (setB.search(temp->rollNo)) result.insert(temp->rollNo);

temp = temp->next;

}

}

void differenceSets(LinkedList& setB, LinkedList& result) {

Node\* temp = head;

while (temp != nullptr) {

if (!setB.search(temp->rollNo)) result.insert(temp->rollNo);

temp = temp->next;

}

}

void display() {

Node\* temp = head;

while (temp != nullptr) {

cout << temp->rollNo << " ";

temp = temp->next;

}

cout << endl;

}

int countNodes() {

int count = 0;

Node\* temp = head;

while (temp != nullptr) {

count++;

temp = temp->next;

}

return count;

}

};

int main() {

LinkedList vanillaSet, butterscotchSet, resultSet;

int totalStudents, rollNo;

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

cin >> totalStudents;

cout << "Enter roll numbers of students who like Vanilla: ";

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

cin >> rollNo;

vanillaSet.insert(rollNo);

}

cout << "Enter roll numbers of students who like Butterscotch: ";

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

cin >> rollNo;

butterscotchSet.insert(rollNo);

}

vanillaSet.unionSets(butterscotchSet, resultSet);

cout << "Set of students who like either Vanilla or Butterscotch or both: ";

resultSet.display();

resultSet = LinkedList();

vanillaSet.intersectionSets(butterscotchSet, resultSet);

cout << "Set of students who like both Vanilla and Butterscotch: ";

resultSet.display();

resultSet = LinkedList();

vanillaSet.differenceSets(butterscotchSet, resultSet);

cout << "Set of students who like only Vanilla: ";

resultSet.display();

resultSet = LinkedList();

butterscotchSet.differenceSets(vanillaSet, resultSet);

cout << "Set of students who like only Butterscotch: ";

resultSet.display();

resultSet = LinkedList();

int neitherCount = totalStudents - vanillaSet.countNodes() - butterscotchSet.countNodes();

cout << "Number of students who like neither Vanilla nor Butterscotch: " << neitherCount << endl;

return 0;

}

Enter total number of students: 5

Enter roll numbers of students who like Vanilla: 101 102 103 104 105

Enter roll numbers of students who like Butterscotch: 103 104 106 107 108

Set of students who like either Vanilla or Butterscotch or both: 105 104 103 102 101 108 107 106

Set of students who like both Vanilla and Butterscotch: 104 103

Set of students who like only Vanilla: 105 102 101

Set of students who like only Butterscotch: 108 107 106

Number of students who like neither Vanilla nor Butterscotch: 0