/\*\*

Second year Computer Engineering class, set A of students like Vanilla Ice-cream and set B of students like butterscotch ice-cream.

Write C/C++ program to store two sets using linked list. compute and display

i. Set of students who like either vanilla or butterscotch or both

ii. Set of students who like both vanilla and butterscotch

iii. Set of students who like only vanilla not butterscotch

iv. Set of students who like only butterscotch not vanilla x.

v. Number of students who like neither vanilla nor butterscotch.

\*\*/

#include <iomanip>

#include <iostream>

using namespace std;

class Student

{

    int rollNo;

    string name;

    Student \*next;

    friend class SecondYear;

public:

    Student()

{

        rollNo=-1;

        name="";

        next=NULL;

}

    Student(int rollNo,string name)

    {

        next=NULL;

        this->rollNo=rollNo;

        this->name=name;

    }

    void printStudentData()

    {

        cout<<endl<<setw(7)<<rollNo<<setw(20)<<name;

    }

};

class SecondYear

{

    Student \*start;

public:

    SecondYear()

{

        start=NULL;

}

    Student\* getStart()

    {

        return start;

    }

    void AddStudent(int rollNo,string name)

    {

        Student \*s=new Student(rollNo,name);

        if(start==NULL)

            start=s;

        else

        {

            s->next=start;

            start=s;

        }

    }

    bool deleteStudent(int rollNo)

    {

        if(start==NULL)

            return false;

        else if(start->next==NULL)

        {

            Student \*temp=start;

            start=NULL;

            delete temp;

            return true;

        }

        else

        {

            Student \*prev=NULL,\*temp=start;

            while(temp!=NULL)

            {

                if(temp->rollNo==rollNo)

                    break;

                prev=temp;

                temp=temp->next;

            }

            if(temp==NULL)

                return false;

            if(temp==start)

                start=start->next;

            else

                prev->next=temp->next;

            delete temp;

            return true;

        }

    }

    void onlySet(SecondYear \*list2)

    {

        Student \*temp2=list2->start;

        Student \*temp1=this->start;

        while(temp1!=NULL)

        {

            bool isPresent=false;

            temp2=list2->start;

            while(temp2!=NULL)

            {

                if(temp1->rollNo==temp2->rollNo)

                {

                    isPresent=true;

                    break;

                }

                temp2=temp2->next;

            }

            if(!isPresent)

                temp1->printStudentData();

            temp1=temp1->next;

        }

    }

    SecondYear\* unionSet(SecondYear \*list2)

    {

        Student \*temp1=start;

        Student \*temp2=list2->start;

        SecondYear \*temp3=new SecondYear;

        while(temp1!=NULL)

        {

            //temp1->printStudentData();

            temp3->AddStudent(temp1->rollNo,temp1->name);

            temp1=temp1->next;

        }

        while(temp2!=NULL)

        {

            bool isPresent=false;

            temp1=start;

            while(temp1!=NULL)

            {

                if(temp1->rollNo==temp2->rollNo)

                {

                    isPresent=true;

                    break;

                }

                temp1=temp1->next;

            }

            if(!isPresent)

            {

                // temp2->printStudentData();

                temp3->AddStudent(temp2->rollNo,temp2->name);

            }

            temp2=temp2->next;

        }

        return temp3;

    }

    void displayNone(SecondYear \*unionSetAB)

    {

        Student \*temp1=start;

        Student \*temp2=unionSetAB->start;

        while(temp1!=NULL)

        {

            bool isPresent=false;

            temp2=unionSetAB->start;

            while(temp2!=NULL)

            {

                if(temp1->rollNo==temp2->rollNo)

                {

                    isPresent=true;

                    break;

                }

                temp2=temp2->next;

            }

            if(!isPresent)

                temp1->printStudentData();

            temp1=temp1->next;

        }

    }

    void intersection(SecondYear \*list2)

    {

        Student \*temp1=start;

        Student \*temp2=NULL;

        while(temp1!=NULL)

        {

            temp2=list2->start;

            while(temp2!=NULL)

            {

                if(temp1->rollNo==temp2->rollNo)

                {

                    temp1->printStudentData();

                    break;

                }

                temp2=temp2->next;

            }

            temp1=temp1->next;

        }

    }

    void displayStudents()

    {

        Student \*temp=start;

        cout<<endl<<setw(7)<<"RollNos" <<setw(20)<<"Names";

        while(temp!=NULL)

        {

            temp->printStudentData();

            temp=temp->next;

        }

    }

};

int main()

{

    SecondYear \*A,\*B,\*universalSet,\*temp=NULL;

    A=new SecondYear;

    B=new SecondYear;

    universalSet=new SecondYear;

    int x=-1,opt,flav;

    int rollNo;

    string name;

    bool del=false;

    do

    {

        cout<<endl;

        cout<<"\*\*\*\*\*\*\*\*\*\*\*\* Second Year Students Volatile Database \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

        cout<<endl<<"1.Enter student\n2.Delete Student\n3.Display students\n4.Display Students(like-Vanilla)";

        cout<<"\n5.Display Students(like-Butterscotch)"<<"\n6.Display Students(like-both Vanilla and ButterScotch)";

        cout<<"\n7.Display Students(like-none of them)\n8.Display Students(like-either Vanilla or ButterScotch or Both)\n9.Exit";

        cin>>opt;

        switch(opt)

        {

        case 1:

            cout<<"\nEnter the name of student ";

            cin>>name;

            cout<<"\nEnter RollNo";

            cin>>rollNo;

            cout<<"\nEnter the flavour liked:\n1.ButterScotch\n2.Vanilla\n3.Both\n4.None";

            cin>>flav;

            switch(flav)

            {

            case 1:B->AddStudent(rollNo,name);

            break;

            case 2:A->AddStudent(rollNo,name);

            break;

            case 3:A->AddStudent(rollNo,name);

            B->AddStudent(rollNo,name);

            break;

            case 4:break;

            default:"Wrong option";

            }

            if(flav>=1 && flav<=4)

                universalSet->AddStudent(rollNo,name);

            break;

            case 2:

                cout<<"Enter rollNo";

                cin>>rollNo;

                del=false;

                if(A->deleteStudent(rollNo))

                {

                    universalSet->deleteStudent(rollNo);

                    del=true;

                }

                if(B->deleteStudent(rollNo))

                {

                    universalSet->deleteStudent(rollNo);

                    del=true;

                }

                if(universalSet->deleteStudent(rollNo))

                    del=true;

                if(!del)

                    cout<<endl<<"No such student";

                else

                    cout<<endl<<"Deleted from database";

                break;

            case 3:

                universalSet->displayStudents();

                break;

            case 4:

                A->onlySet(B);

                break;

            case 5:

                B->onlySet(A);

                break;

            case 6:

                A->intersection(B);

                break;

            case 7:

                temp=A->unionSet(B);

                universalSet->displayNone(temp);

                delete temp;

                break;

            case 8:temp=A->unionSet(B);

            temp->displayStudents();

            delete temp;

            break;

            case 9:break;

            default:cout<<"Wrong option";

            break;

        }

    }while(x!=9);

}