//========================================================================

// Name : 21465\_Pract7.cpp

// Author : Chaitanya Paraskar

// Roll No. : 21465

// Aim : Write C++ program for storing binary number using doubly linked lists.

// Write functions- a) To compute 1's and 2's complement

// b) Add two binary numbers

//========================================================================

#include "iostream"

using namespace std;

class Node

{

private:

int data;

Node \*next;

friend class Number;

};

class Number

{

private:

Node \*start;

public:

Number(string num);

void display();

void reverseDisplay(Node \*ptr);

Number \*add(Number \*n2);

void onecomp();

void twocomp();

};

Number::Number(string num)

{

for (int i = 0; i < num.length(); i++)

{

char ch = num[i];

Node \*n = new Node();

if (ch == '0')

n->data = 0;

else

n->data = 1;

n->next = this->start;

this->start = n;

}

}

void Number::display()

{

// Node \*ptr = this->start;

// cout << "Number -> ";

// while (ptr != NULL)

// {

// cout << ptr->data << " -> ";

// ptr = ptr->next;

// }

// cout << "NULL" << endl;

this->reverseDisplay(this->start);

cout << endl;

}

void Number::reverseDisplay(Node \*ptr)

{

if (ptr->next != NULL)

{

this->reverseDisplay(ptr->next);

cout << ptr->data;

}

else

{

cout << ptr->data;

}

}

void Number::onecomp()

{

Node \*ptr = this->start;

while (ptr != NULL)

{

if (ptr->data == 0)

ptr->data = 1;

else

ptr->data = 0;

ptr = ptr->next;

}

}

void Number::twocomp()

{

cout << "After One's Complement :-" << endl;

this->onecomp();

cout << "n1 => ";

this->display();

Node \*ptr = this->start;

int carry = 1;

while (ptr != NULL)

{

if (ptr->data == 0)

{

ptr->data = carry;

carry = 0;

}

if (ptr->data == 1 && carry == 0)

{

}

if (ptr->data == 1 && carry == 1)

{

ptr->data = 0;

carry = 1;

}

ptr = ptr->next;

}

if (carry == 1)

{

Node \*n = new Node();

n->data = 1;

n->next = this->start;

this->start = n;

}

}

Number \*Number::add(Number \*n)

{

string str = "";

int carry = 0;

Node \*n1 = this->start;

Node \*n2 = n->start;

while (n1 != NULL && n2 != NULL)

{

if (n1->data == 0 && n2->data == 0)

{

if (carry == 0)

str = "0" + str;

else

str = "1" + str;

carry = 0;

}

if (n1->data == 0 && n2->data == 1)

{

if (carry == 0)

{

str = "1" + str;

carry = 0;

}

else

{

str = "0" + str;

carry = 1;

}

}

if (n1->data == 1 && n2->data == 0)

{

if (carry == 0)

{

str = "1" + str;

carry = 0;

}

else

{

str = "0" + str;

carry = 1;

}

}

if (n1->data == 1 && n2->data == 1)

{

if (carry == 0)

{

str = "0" + str;

carry = 1;

}

else

{

str = "1" + str;

carry = 1;

}

}

n1 = n1->next;

n2 = n2->next;

}

if (carry == 1)

{

str = "1" + str;

}

Number \*res = new Number(str);

return res;

}

int main()

{

Number \*n1 = new Number("101101");

Number \*n2 = new Number("001101");

cout << "n1 => ";

n1->display();

cout << "n2 => ";

n2->display();

Number \*n3 = n1->add(n2);

cout << "n3 => ";

n3->display();

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "n1 => ";

n1->display();

cout << "After One's Complement :-" << endl;

n1->onecomp();

cout << "n1 => ";

n1->display();

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "n1 => ";

n1->display();

n1->twocomp();

cout << "After Two's Complement :-" << endl;

cout << "n1 => ";

n1->display();

return 0;

}

/\*

Output:

$ g++ Pract7Binary.cpp -o out && ./out

n1 => 101101

n2 => 001101

n3 => 111010

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

n1 => 101101

After One's Complement :-

n1 => 010010

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

n1 => 010010

After One's Complement :-

n1 => 101101

After Two's Complement :-

n1 => 101110

\*/

//========================================================================

// Name : 21465\_Pract8.cpp

// Author : Chaitanya Paraskar

// Roll No. : 21465

// Aim : 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 -

// a) Set of students who like both vanilla and butterscotch

// b) Set of students who like either vanilla or butterscotch or not both

// c) Number of students who like neither vanilla nor butterscotch

//========================================================================

#include "iostream"

using namespace std;

class Student

{

public:

int rno;

Student \*next;

Student(int rno)

{

this->rno = rno;

}

};

class Set

{

Student \*head;

int count = 0;

void add(int rno)

{

Student \*s = new Student(rno);

s->next = this->head;

head = s;

count++;

}

friend Set \*initUniversal();

public:

Set() {}

Set(Set \*R)

{

int n;

cout << "No. of Students : ";

cin >> n;

int i = 0;

while (i < n)

{

int rno;

cout << "Enter Roll No. of " << i + 1 << "th student : ";

cin >> rno;

if (this->not\_in(rno) && R->in(rno))

{

this->add(rno);

i++;

}

else

{

cout << "Already Present in Set !!" << endl;

}

}

}

void display()

{

cout << "Set : [";

Student \*ptr = this->head;

while (ptr != NULL)

{

cout << ptr->rno;

if (ptr->next != NULL)

cout << ", ";

ptr = ptr->next;

}

cout << "]" << endl;

}

bool in(int n)

{

Student \*ptr = this->head;

while (ptr != NULL)

{

if (ptr->rno == n)

return true;

ptr = ptr->next;

}

return false;

}

bool not\_in(int n)

{

Student \*ptr = this->head;

while (ptr != NULL)

{

if (ptr->rno == n)

return false;

ptr = ptr->next;

}

return true;

}

int getCount()

{

return this->count;

}

static Set \*And(Set \*A, Set \*B);

static Set \*Or(Set \*A, Set \*B);

static Set \*Difference(Set \*A, Set \*B);

};

Set \*Set::And(Set \*A, Set \*B)

{

Set \*R = new Set();

Student \*a = A->head;

while (a != NULL)

{

if (B->in(a->rno))

R->add(a->rno);

a = a->next;

}

return R;

}

Set \*Set::Or(Set \*A, Set \*B)

{

Set \*R = new Set();

Student \*a = A->head;

while (a != NULL)

{

if (R->not\_in(a->rno))

R->add(a->rno);

a = a->next;

}

Student \*b = B->head;

while (b != NULL)

{

if (R->not\_in(b->rno))

R->add(b->rno);

b = b->next;

}

return R;

}

Set \*Set::Difference(Set \*A, Set \*B)

{

Set \*R = new Set();

Student \*a = A->head;

while (a != NULL)

{

if (B->not\_in(a->rno))

R->add(a->rno);

a = a->next;

}

return R;

}

Set \*initUniversal()

{

cout << "Enter Universal Set : " << endl;

Set \*U = new Set();

int n;

cout << "No. of Students : ";

cin >> n;

int i = 0;

while (i < n)

{

int rno;

cout << "Enter Roll No. of " << i + 1 << "th student : ";

cin >> rno;

if (U->not\_in(rno))

{

U->add(rno);

i++;

}

else

{

cout << "Already Present in Set !!" << endl;

}

}

return U;

}

int main()

{

Set \*U = initUniversal();

cout << "Enter Set of Roll No. of Students who like Vanilla : " << endl;

Set \*Vanilla = new Set(U);

cout << "Enter Set of Roll No. of Students who like Butter Scotch : " << endl;

Set \*ButterScotch = new Set(U);

cout << "Roll No. of Students who Like Vanilla : " << endl;

Vanilla->display();

cout << "Roll No. of Students who Like Butter Scotch : " << endl;

ButterScotch->display();

cout << "Students who like both Vanilla and Butter Scotch : " << endl;

Set \*a = Set::And(Vanilla, ButterScotch);

a->display();

cout << "Students who like Either Vanilla or Butter Scotch but not both : " << endl;

Set \*x = Set::Or(Vanilla, ButterScotch);

Set \*y = Set::And(Vanilla, ButterScotch);

Set \*b = Set::Difference(x, y);

b->display();

Set \*c = Set::Difference(U, x);

int count = c->getCount();

cout << "No. of Students who like neither Vanilla, nor Butter Scotch : " << count << endl;

return 0;

}

/\*

Output:

$ g++ Pract8.cpp -o out && ./out

Enter Universal Set :

No. of Students : 4

Enter Roll No. of 1th student : 111

Enter Roll No. of 2th student : 222

Enter Roll No. of 3th student : 333

Enter Roll No. of 4th student : 444

Enter Set of Roll No. of Students who like Vanilla :

No. of Students : 2

Enter Roll No. of 1th student : 111

Enter Roll No. of 2th student : 222

Enter Set of Roll No. of Students who like Butter Scotch :

No. of Students : 2

Enter Roll No. of 1th student : 111

Enter Roll No. of 2th student : 333

Roll No. of Students who Like Vanilla :

Set : [222, 111]

Roll No. of Students who Like Butter Scotch :

Set : [333, 111]

Students who like both Vanilla and Butter Scotch :

Set : [111]

Students who like Either Vanilla or Butter Scotch but not both :

Set : [222, 333]

No. of Students who like neither Vanilla, nor Butter Scotch : 1

\*/