```
//============================================================================
// 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

*/
```