#include<iostream> using namespace std; class binary;

class node

{

public:

node \*prev; bool n; node\*next;

node()

{

prev=next=NULL;

}

node(bool b)

{

n=b; prev=next=NULL;

}

friend class binary;

};

class binary

{

node \*start;

public:

binary()

{

start=NULL;

}

void generateBinary(int no); void displayBinary();

void onesComplement(); void twoscomplement();

binary operator +(binary n1); bool addBitAtBegin(bool val)

{

node \*nodee=new node(val); if(start==NULL)

{

}

else

{

}

start=nodee;

nodee->next=start; start->prev=nodee; start=nodee;

return true;

}

};

void binary::generateBinary(int no)

{

bool rem; node \*p; rem=no%2;

start=new node(rem); no=no/2; while(no!=0)

{

rem=no%2; no=no/2;

/\*

if(start==NULL)

{

start=new node(rem);

// cout<<" Start prev: "<<start->prev;

// cout<<" Start next: "<<start->next ;

}

else

{

\*/

p=new node(rem); p->next=start; start->prev=p;

// cout<<" Start prev: "<<start->prev->n;

// cout<<" p->n"<<p->n; start=p;

//}

}

}

void binary::displayBinary()

{

node \*t; t=start; while(t!=NULL)

{

cout<<t->n; t=t->next;

}

}

void binary::onesComplement()

{

node \*t; t=start;

while(t!=NULL)

{

if(t->n==0)

t->n=1;

else

t->n=0;

t=t->next;

}

}

binary binary::operator +(binary n1)

{

binary sum; node \*a=start;

node \*b=n1.start;

// bit \*s=sum.start; bool carry=false; while(a->next!=NULL)

a=a->next; while(b->next!=NULL)

b=b->next;

while(a!=NULL && b!=NULL)

{

sum.addBitAtBegin((a->n)^(b->n)^carry);

carry=((a->n&& b->n) || (a->n&& carry) || (b->n && carry));

a=a->prev; b=b->prev;

}

while(a!=NULL)

{

sum.addBitAtBegin(a->n^carry); a=a->prev;

}

while(b!=NULL)

{

sum.addBitAtBegin(b->n^carry); b=b->prev;

}

sum.addBitAtBegin(carry); return sum;

}

void binary::twoscomplement()

{

onesComplement(); bool carry=1; node \*t;

t=start;

while(t->next!=NULL)

{

t=t->next;

}

while(t!=NULL)

{

if(t->n==1&& carry==1)

{

}

else

t->n=0;

carry=1;

if(t->n==0&& carry==1)

{

}

else

t->n=1;

carry=0;

if(carry==0) break;

t=t->prev;

}

displayBinary();

}

int main()

{

int num,num1; binary n1,n3,n2; int choice=1;

do

{

cout<<"\n\n=========Binary Number Operations========\n";

cout<<"1. Generate binary\n2.One's Complement\n3.Two's Complement\n4.

Addition\n0.Exit\nEnter your choice: ";

cin>>choice; switch(choice)

{

case 1: cout<<"\nENter Number in decimal form: ";

cin>>num; n1.generateBinary(num);

cout<<"\nBinary Representation: "; n1.displayBinary();

break;

case 2:cout<<"\nENter Number in decimal form: ";

cin>>num; n1.generateBinary(num);

cout<<"\nBinary Representation: "; n1.displayBinary();

cout<<"\nOnes Complement: "; n1.onesComplement(); n1.displayBinary();

break;

case 3:cout<<"\nENter Number in decimal form: ";

cin>>num; n1.generateBinary(num);

cout<<"\nBinary Representation: "; n1.displayBinary();

cout<<"\nTwos complement; "; n1.twoscomplement();

break;

case 4: cout<<"\nENter Two Numbers: ";

cin>>num>>num1; n1.generateBinary(num); n2.generateBinary(num1); n1.displayBinary(); cout<<" + "; n2.displayBinary(); cout<<"= ";

n3=n1+n2; n3.displayBinary();

}

}while(choice!=0); n1.generateBinary(7); cout<<"\nBinary Representation: "; n1.displayBinary();

n1.displayBinary(); cout<<"\nTwos complement; "; n1.twoscomplement();

return 0;

}