CODE1:

#include <iostream>

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

using namespace std;

struct Employees{

string id;

string name;

Employees \* next;

};

void addEmp(Employees \*\*head,string s1,string s2){

Employees \*newNode = new Employees();

Employees \*lastVal = \*head;

newNode->id = s1;

newNode->name = s2;

newNode->next = NULL;

if(\*head == NULL)

{

\*head = newNode;

return;

}

else

{

while(lastVal->next!=NULL)lastVal = lastVal->next;

lastVal->next = newNode;

return ;

}

}

void retire(Employees \*\* head, string s1,string s2){

Employees \*sub = \*head;

Employees \*prev = NULL;

if(sub!=NULL && sub->name==s2){

\*head = sub->next;

delete sub;

return ;

}

else{

while(sub!=NULL && sub->name!=s2){

prev = sub;

sub = sub->next;

}

if(sub == NULL) return;

prev->next = sub->next;

delete sub;

return;

}

}

void listRev(Employees\* node){

if(node==NULL)return;

else listRev(node->next);

cout<<node->name<<'\n';

}

int main(){

Employees \*Head = NULL;

addEmp(&Head,"A1","Omkar Bharitkar");

addEmp(&Head,"A2","Yash Bharitkar");

addEmp(&Head,"A3","Rajendra Bharitkar");

addEmp(&Head,"A4","Aarya Bharitkar");

addEmp(&Head,"A5","Minakshi Bharitkar");

int count = 0;

Employees \*head = Head;

while(head!=NULL){

count++;

head = head->next;

}

cout<<"Total number of employees working in this industry: "<< count <<"\n\n";

head = Head;

cout<<" \* \* List of the Employees \* \*"<<'\n';

cout<<" ID Name "<<'\n';

while(head!=NULL){

cout<<(head->id)[0]<<") "<<head->id<<" "<<head->name<<'\n';

head = head->next;

}

addEmp(&Head,"A6","Atharv Bharitkar");

cout<< endl;

cout<<"The name of new employee hired is: ";

head = Head ;

while(head!=NULL){

if(head->next==NULL)

{

cout<<head->name<<"\n\n";

}

head = head->next;

}

head = Head;

retire(&Head,"A2","Yash Bharitkar");

cout<<"One of our employee retired. The list of remaining employees is: "<<"\n";

count =0 ;

while(head!=NULL){

count++;

cout<<head->name<<'\n';

head = head->next;

}

cout << endl;

cout<<"No. of Employees: "<< count <<"\n\n";

cout<<"List of employees in reverse:\n";

listRev(Head);

cout<<"\nEmployees in the other department are:\n";

Employees\* Head1 = NULL;

addEmp(&Head1,"B2","Nitesh Bharitkar");

addEmp(&Head1,"B3","Shruti Bharitkar");

addEmp(&Head1,"B4","Sagar Bharitkar");

addEmp(&Head1,"B5","Pallavi Bharitkar");

head = Head1;

while(head!=NULL){

cout<<head->id<<" "<<head->name<<'\n';

head = head->next;

}

cout<<"\nMerging both the departments, the new list of employees will be:\n";

head = Head;

while(true){

if(head->next == NULL){

head->next = Head1;

break;

}

head = head->next;

}

head = Head;

while(head!=NULL){

cout<<head->id<<" "<<head->name<<"\n";

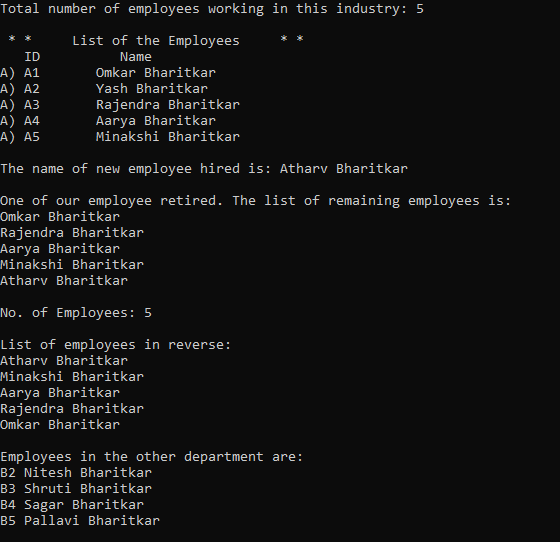
head = head->next;

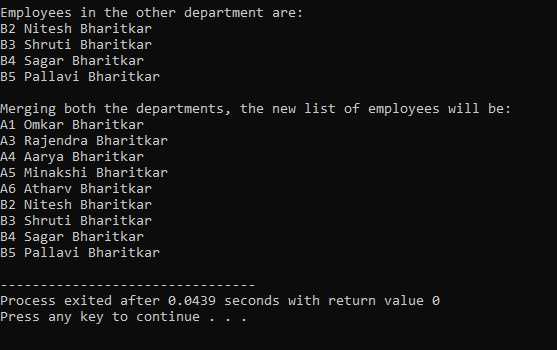
}

return 0;

}

Output:





CODE2:

#include <iostream>

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

using namespace std;

class binary;

class node

{

node \*prev;

bool n;

node\*next;

public:

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)

{

start=nodee;

}

else

{

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)

{

t->n=0;

carry=1;

}

else

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

{

t->n=1;

carry=0;

}

else

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();

//

// cout<<"\nOnes Complement: ";

// n1.displayBinary();

cout<<"\nTwos complement; ";

n1.twoscomplement();

return 0;

}

Output:

