```
/*Write double linked list in C++
Implement the following functions
1. addToBack
2. addToFront
3. printForward
4. printBackward
5. remove
6. insertBefore
7. insertAfter
8. removeAll*/
#include <iostream>
using namespace std;

class node
{
public:
    node *prev;
    int value;
    node *next;
};

node * head;
node * tail;
    int num;


class doublelink
{
public:
    void PrintBackword()
    {
        cout<<"print Backword"<<endl;
    for(node *current=tail; current!=nullptr; current=current->prev)
    {
        cout << current->value  << endl;

    }
    }

    void PrintForword()
    {
        cout<<"print Forword"<<endl;
    for(node * current=head; current!=nullptr; current=current->next)
    {
        cout << current->value  << endl;
    }

    }

    void AddToBack(int num)
    {
        cout<<"after adding element to Back"<<endl;

            node *newnode=new node;
            newnode->value=num;
            newnode->next=nullptr;
            newnode->prev = nullptr;

        if (nullptr == head)
        {
            head = newnode;
        }
        else
        {
            newnode->prev=tail;
            tail->next=newnode;
```

```
            newnode->next=nullptr;
        }
        tail = newnode;
    }
    void AddToFront(int num)
    {
        cout<<"After adding element to front "<<endl;
        node *newnode=new node;
        newnode->value=num;
        newnode->next=nullptr;
        newnode->prev=nullptr;

        if(nullptr == head)
        {
            tail=newnode;
        }
        else
        {
            newnode->next=head;
            head->prev=newnode;
            newnode->prev=nullptr;
        }
        head=newnode;
    }
    void InsertAfter(int num)
    {
        node *newnode=new node;
        newnode->value=num;
        newnode->next=nullptr;
        newnode->prev=nullptr;

        if(nullptr == head)
        {
            head=newnode;
        }
        else
        {

        }
        tail=newnode;
    }

    int RemoveAll()
    {
        cout<<"remove all"<<endl;
        int cnt=0;

        while(head!=0)
        {
            node *temp;
            temp = head;
            head = head->next;
            delete temp;
        }
        cnt++;
        tail = nullptr;
        return cnt;
    }

    bool remove()
    {
        cout<<"Enter number to delete"<<"\t"<<endl;
        cin>>num;
        for(node *p;p=nullptr;p=p->next)
        {
```

```
            if(num=p->value)
            {
                delete p;
                p->next->prev=p->prev;
                p->prev->next=p->next;
                return true;
            }
            else
            {
            cout<<"not found";
            }
        return false;

        }
    }

};

int main()
{
    int num;
    while(cout<<"Insert or 0 to exit :", cin >> num, num!=0)
    {
        node *newnode=new node;
        newnode->value=num;
        newnode->next=nullptr;
        newnode->prev=nullptr;

        if(nullptr == head)
        {
            head=newnode;
        }
        else
        {
                tail->next = newnode;
                newnode->prev = tail;
                newnode->next = nullptr;
        }
        tail=newnode;
    }

    doublelink d;

    d.PrintForword();

    d.PrintBackword();

    d.AddToFront(15);
    d.PrintForword();

    d.AddToBack(100);
    d.PrintForword();

    d.remove();
    d.PrintForword();

//  d.RemoveAll();
    //d.PrintForword();
}
```