i. Insert a new node at the end of the list.

```
void add_node_tail(int n){
//add to tail
        if(tail!=NULL)
    {
        tail= new node(n,NULL,tail);
        tail->prev->next=tail;
    }
    else{
        head=tail=new node(n);
    }
}
```

ii. Insert a new node at the beginning of list.

```
void add_node_head(int n)
{
    if(head == NULL) {
        head=tail=new node(n,NULL,NULL);
    }
        else{
            head=new node(n,head,NULL);
            head->next->prev=head;
        }
}
```

iii. Insert a new node at given position.

```
void add_somewhere(int n, int a){
    if(head==tail){
        head=tail=new node(n,NULL,NULL);
    }
    else{
        node *tmp, *place;
        for(tmp=head;tmp!=0;tmp=tmp->next){
            if(tmp->data==a){
            place=new node(n,tmp->next,tmp);
            if(tmp->next!=NULL){
                tmp->next->prev=place;
        }
        else{
            tail=place;
    }
        tmp->next=place;

        }

    }
}
}
```

iv. Delete any node.

```
void delete_head() { //delete first node
    if (head!=NULL){
        int delnode = head->data;
        node *tmp = head;
    if (head == tail){
        head = tail = NULL;

    }
    else{
        head = head->next;
        delete tmp;
        head->prev=NULL;
    }
}
    else{
        cout<<"list is empty";
    }
}
void delete_tail() { //delete last node
        int delnode = tail->data;
        if (head == tail) {
        delete head;
        head=tail=NULL;
    }
        else{
            tail=tail->prev;
            delete tail->next;
            tail->next=NULL;
    }
}
```

v. Print the complete doubly link list.

```
void display()
{
    node *temp=new node;
    temp=head;
    while(temp!=NULL) {
        cout<<temp->data<<endl;
        temp=temp->next;
    }
}
```

Task-2:
Create a circular link list and perform the mentioned tasks.

i. Insert a new node at the end of the list.

```
void insert_end(int n)   {
   node *tmp;
   tmp=new node;
   tmp->data=n;

  node *t=head;
   tmp->next = NULL;

   if (head == NULL) {
      head = tmp;
   }
   while (t->next != NULL)
   t = t->next;

   t->next = tmp;

}
```

```
    void front(int n){
        node *tmp;
        tmp=new node;
        tmp->data=n;
        if(head==NULL){
          head=tmp;
        }
        else{
          tmp->next=head;
          head=tmp;
          node *t;
          t=head;
          while(t!=NULL){

              t=t->next;
          }
          t->next=head;




    }
}
```

iii. Insert a new node at given position.

```
        void insert_at_any_position(int num, int pos){
                node *tmp, *t;
                int i;


                if(pos == 1){
                    front(num);
                }
                else{
                    tmp = new node;
                    tmp->data = num;
                    t = head;
                    for(i=2; i<=pos-1; i++){
                        t = t->next;
                    }
                tmp->next = t->next;
                t->next = tmp;
    }
}
```

iv. Delete any node.

```
        void delete_head() { //delete first node
            if (head!=NULL){
            int delnode = head->data;
            node *tmp = head;
                    if (head == tail){
                            head = tail = NULL;}
                    else{
                        head = head->next;
                        delete tmp;
                        head->pre=NULL;
            }
}
        else{
            cout<<"list is empty";
        }
}
```

v. Print the complete circular link list.

```
        void Display() {
            // struct Node* last;
            node *t;
            node *tmp;
            tmp=head;
            while (tmp != NULL) {
                cout<<tmp->data<<" ";
                t = tmp;
                tmp = tmp->next;
                    }

    }
```