Q1.# Insertion and deletion in singly linkedlist:

```
#include<iostream>

using namespace std;

class Node {
private:
   int data;
   Node* next;

public:
   Node* head;

   Node() {
      head = NULL;
   }

  void insert_end(int n){

                    if(head==NULL)
                    {

                     head=new Node();
                     head->data=n;
                     head->next=NULL;

                    }

                    else
```

```
        {

          Node *p,*ptr;
          ptr=head;
          while(ptr->next!=NULL)
          {
                ptr=ptr->next;
          }

          p=new Node();
          p->data=n;
          p->next= NULL;
          ptr->next=p;

        }
}

        void insert_beg(int n){

        if(head==NULL)
        {

         head=new Node();
         head->data=n;
         head->next=NULL;

        }

        else

        {
```

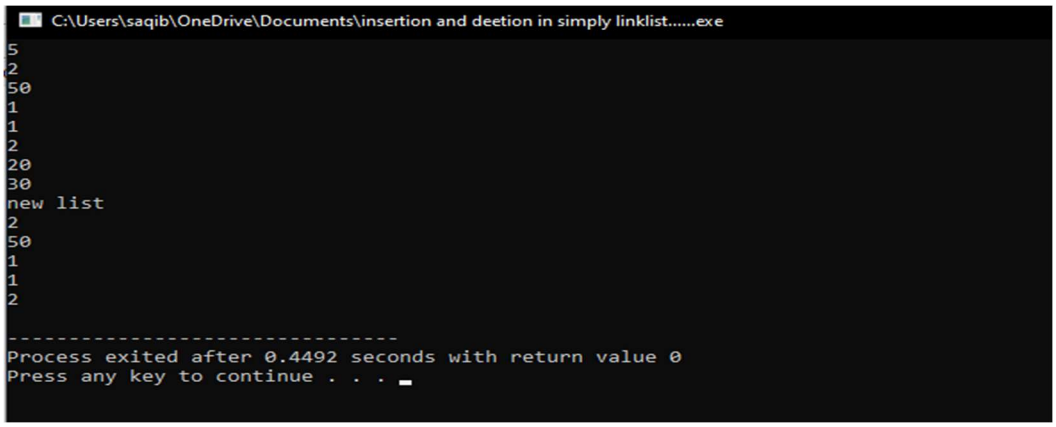
```
            Node *p;
            p=new Node();
            p->data=n;
            p->next= head;
            head=p;

          }
}

void insert_at_value(int pos,int n){

          if(head==NULL)
          {

           head=new Node();
           head->data=n;
           head->next=NULL;

          }

          else

          {
            Node *ptr;
            ptr=head;
            while(ptr->data!=pos)
            {
            ptr=ptr->next;

                    }
```

```
                Node *p;
                p=new Node();
                p->data=n;
                p->next= ptr->next;
                ptr->next=p;
                }

            }

        void display()
        {
        Node *ptr;
        ptr=head;
        if(ptr==NULL)
        {
            cout << " \nNo data is in the list.."<<endl;
            return;
        }
        else{

            while(ptr!=NULL){
                cout<<ptr->data<<endl;
                ptr=ptr->next;
            }
        }

}



void delete_beg() {
```

```
    if (head == NULL) {
        cout << "List is empty. Cannot delete." << endl;
        return;
    }

    Node* temp = head;
    head = head->next;
    delete temp;
}

void delete_end() {
    if (head == NULL) {
        cout << "List is empty. Cannot delete." << endl;
        return;
    }

    if (head->next == NULL) {
        delete head;
        head = NULL;
        return;
    }

    Node* ptr = head;
    while (ptr->next->next != NULL) {
        ptr = ptr->next;
    }

    delete ptr->next;
    ptr->next = NULL;
}
```

```
    void delete_at_value(int val) {
        if (head == NULL) {
            cout << "List is empty. Cannot delete." << endl;
            return;
        }

        if (head->data == val) {
            Node* temp = head;
            head = head->next;
            delete temp;
            return;
        }

        Node* ptr = head;
        while (ptr->next != NULL && ptr->next->data != val) {
            ptr = ptr->next;
        }

        if (ptr->next != NULL) {
            Node* temp = ptr->next;
            ptr->next = ptr->next->next;
            delete temp;
        } else {
            cout << "Value not found in the list." << endl;
        }
    }

    // ... (display method remains the same)
};

int main() {
```

```
    Node n;
    n.insert_beg(1);
    n.insert_beg(2);
    n.insert_end(1);
    n.insert_end(2);
    n.insert_end(20);
    n.insert_at_value(2, 50);
    n.insert_end(30);
    n.insert_beg(5);

    n.display();
        cout<<"new list"<<endl;
    n.delete_beg();
    n.delete_end();
    n.delete_at_value(20);

    n.display();

    return 0;
}
```



Q2.# insertion and deletion doubly linkedlist:

```
#include <iostream>

using namespace std;

class Node {
private:
    int data;
    Node* next;
    Node* prev;

public:
    Node* head;

    Node() {
        head = NULL;
    }

    void insert_end(int n) {
        if (head == NULL) {
            head = new Node();
```

```
        head->data = n;
        head->next = NULL;
        head->prev = NULL;
    } else {
        Node* p, * ptr;
        ptr = head;
        while (ptr->next != NULL) {
            ptr = ptr->next;
        }

        p = new Node();
        p->data = n;
        p->next = NULL;
        p->prev = ptr;
        ptr->next = p;
    }
}

void insert_beg(int n) {
    if (head == NULL) {
        head = new Node();
        head->data = n;
```

```
        head->next = NULL;
        head->prev = NULL;
    } else {
        Node* p;
        p = new Node();
        p->data = n;
        p->next = head;
        p->prev = NULL;
        head->prev = p;
        head = p;
    }
}

void insert_at_value(int pos, int n) {
    if (head == NULL) {
        head = new Node();
        head->data = n;
        head->next = NULL;
        head->prev = NULL;
    } else {
        Node* ptr;
        ptr = head;
```

```
        while (ptr->data != pos) {
            ptr = ptr->next;
        }

        Node* p;
        p = new Node();
        p->data = n;
        p->next = ptr->next;
        p->prev = ptr;
        if (ptr->next != NULL) {
            ptr->next->prev = p;
        }
        ptr->next = p;
    }
}

void delete_beg() {
    if (head == NULL) {
        cout << "List is empty. Cannot delete." << endl;
        return;
    }
```

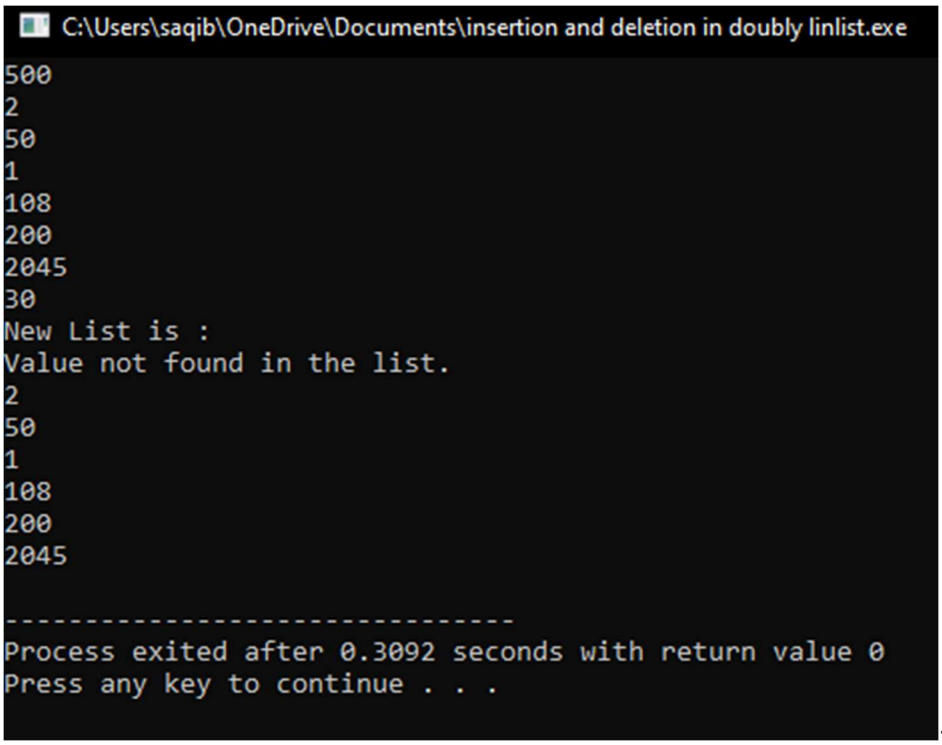
```
    Node* temp = head;
    head = head->next;
    if (head != NULL) {
        head->prev = NULL;
    }
    delete temp;
}

void delete_end() {
    if (head == NULL) {
        cout << "List is empty. Cannot delete." << endl;
        return;
    }

    Node* ptr = head;
    while (ptr->next != NULL) {
        ptr = ptr->next;
    }

    if (ptr->prev != NULL) {
        ptr->prev->next = NULL;
    } else {
```

```
        head = NULL;
    }
    delete ptr;
}

void delete_at_value(int val) {
    if (head == NULL) {
        cout << "List is empty. Cannot delete." << endl;
        return;
    }

    Node* ptr = head;
    while (ptr != NULL && ptr->data != val) {
        ptr = ptr->next;
    }

    if (ptr == NULL) {
        cout << "Value not found in the list." << endl;
        return;
    }

    if (ptr->prev != NULL) {
```

```
        ptr->prev->next = ptr->next;
    } else {
        head = ptr->next;
    }

    if (ptr->next != NULL) {
        ptr->next->prev = ptr->prev;
    }

    delete ptr;
}

void display() {
    Node* ptr;
    ptr = head;
    if (ptr == NULL) {
        cout << "\nNo data is in the list.." << endl;
        return;
    } else {
        while (ptr != NULL) {
            cout << ptr->data << endl;
            ptr = ptr->next;
```

```
            }
        }
    }
};

int main() {
    Node n;
    n.insert_beg(1);
    n.insert_beg(2);
    n.insert_end(108);
    n.insert_end(200);
    n.insert_end(2045);
    n.insert_at_value(2, 50);
    n.insert_end(30);
    n.insert_beg(500);

    n.display();
        cout<<"New List is :"<<endl;
    n.delete_beg();
    n.delete_end();
    n.delete_at_value(20);
```

```
    n.display();

    return 0;
}
```



#

Q3#.Insertion and deletion of doubly linkedlist:

```
#include <iostream>

using namespace std;

class Node {
private:
```

```
    int data;
    Node* next;

public:
    Node* head;

    Node() {
        head = NULL;
    }

    void insert_end(int n) {
        if (head == NULL) {
            head = new Node();
            head->data = n;
            head->next = head;  // Circular: Point to itself
        } else {
            Node* p, * ptr;
            ptr = head;
            while (ptr->next != head) {
                ptr = ptr->next;
            }
```

```
        p = new Node();
        p->data = n;
        p->next = head;
        ptr->next = p;
    }
}

void insert_beg(int n) {
    if (head == NULL) {
        head = new Node();
        head->data = n;
        head->next = head;  // Circular: Point to itself
    } else {
        Node* p, * ptr;
        ptr = head;
        while (ptr->next != head) {
            ptr = ptr->next;
        }

        p = new Node();
        p->data = n;
        p->next = head;
```

```
        ptr->next = p;
        head=p;
    }
}

void insert_at_value(int pos, int n) {
    if (head == NULL) {
        cout << "List is empty. Cannot insert." << endl;
        return;
    }

    Node* ptr;
    ptr = head;
    while (ptr->data != pos) {
        ptr = ptr->next;
    }

    Node* p;
    p = new Node();
    p->data = n;
    p->next = ptr->next;
    ptr->next = p;
```

```
    }

void display() {
    if (head == NULL) {
        cout << "No data is in the list." << endl;
        return;
    }

    Node* ptr = head;
    do {
        cout << ptr->data << endl;
        ptr = ptr->next;
    } while (ptr != head);
}



    void delete_beg() {
        if (head == NULL) {
            cout << "List is empty. Cannot delete." << endl;
            return;
        }
```

```
    Node* temp = head;
    Node* ptr = head;
    while (ptr->next != head) {
        ptr = ptr->next;
    }
    head = head->next;
    ptr->next = head;
    delete temp;
}

void delete_end() {
    if (head == NULL) {
        cout << "List is empty. Cannot delete." << endl;
        return;
    }

    if (head->next == head) {
        delete head;
        head = NULL;
        return;
    }
```

```
    Node* ptr = head;
    Node* prev = nullptr;
    while (ptr->next != head) {
        prev = ptr;
        ptr = ptr->next;
    }

    prev->next = head;
    delete ptr;
}

void delete_at_value(int val) {
    if (head == NULL) {
        cout << "List is empty. Cannot delete." << endl;
        return;
    }

    if (head->data == val) {
        Node* temp = head;
        Node* ptr = head;
        while (ptr->next != head) {
            ptr = ptr->next;
```

```
        }
        head = head->next;
        ptr->next = head;
        delete temp;
        return;
    }

    Node* ptr = head;
    Node* prev = nullptr;
    do {
        prev = ptr;
        ptr = ptr->next;
    } while (ptr != head && ptr->data != val);

    if (ptr != head) {
        prev->next = ptr->next;
        delete ptr;
    } else {
        cout << "Value not found in the list." << endl;
    }
  }
};
```

```
int main() {
    Node n;
    n.insert_beg(19);
   n.insert_beg(2);
    n.insert_beg(4);
    n.insert_end(1);
    n.insert_end(2);
    n.insert_end(20);
    n.insert_at_value(2, 50);
    n.insert_end(30);
    n.insert_beg(5);

    n.display();
    cout << "Hi New list nicha ha" << endl;
    n.delete_beg();
    n.delete_end();
    n.delete_at_value(20);

    n.display();

    return 0;
```

}