

ASSISMENT 2

Student Name: Bhaskar Kumar

UID: 23BCS14337

Branch:BE-CSE

Section/Group: KRG-3B

Semester: 5

Date of Performance:24-07-25

Subject Name: DAA

Subject Code:23CSH-301

1. **Aim:** WAP of Liked List(singly), Doubly Linked List, Circular Linked List insertion and deletion for each .
2. **Objective:** To develop a program showing insertion and deletion in a linked list (using C++) and the algorithm analysis of
->Insertion at begin, end or any given position for singly, doubly, circular linked list.
->Deletion at begin, end or any given position singly, doubly, circular linked list.

3. CODE:

```
#include <iostream>
```

```
using namespace std;
```

```
// ----- Singly Linked List -----
```

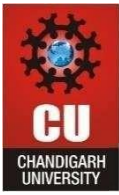
```
struct SNode {
```

```
    int data;
```

```
    SNode* next;
```

```
    SNode(int val) : data(val), next(NULL) {}
```

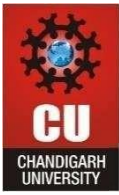
```
};
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
class SinglyList {  
  
    SNode* head = NULL;  
  
public:  
  
    void insert(int pos, int val) {  
  
        SNode* newNode = new SNode(val);  
  
        if (pos == 1 || !head) {  
  
            newNode->next = head;  
  
            head = newNode;  
  
            return;  
  
        }  
  
        SNode* temp = head;  
  
        for (int i = 1; i < pos - 1 && temp->next; ++i)  
  
            temp = temp->next;  
  
        newNode->next = temp->next;  
  
        temp->next = newNode;  
  
    }  
  
    void remove(int pos) {  
  
        if (!head) return;  
  
        if (pos == 1) {  
  
            SNode* del = head;  
  
            head = head->next;  
  
            delete del;  
  
            return;  
  
        }  
  
    }  
  
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
    }

    SNode* temp = head;

    for (int i = 1; i < pos - 1 && temp->next; ++i)

        temp = temp->next;

    if (!temp->next) return;

    SNode* del = temp->next;

    temp->next = del->next;

    delete del;

}
```

```
void print() {

    SNode* temp = head;

    cout << "Singly List: ";

    while (temp) {

        cout << temp->data << " -> ";

        temp = temp->next;

    }

    cout << "NULL\n";

}

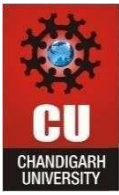
};
```

// ----- Doubly Linked List -----

```
struct DNode {

    int data;

    DNode* prev;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
DNode* next;
```

```
DNode(int val) : data(val), prev(NULL), next(NULL) {}
```

```
};
```

```
class DoublyList {
```

```
    DNode* head = NULL;
```

```
    DNode* tail = NULL;
```

```
public:
```

```
    void insert(int pos, int val) {
```

```
        DNode* newNode = new DNode(val);
```

```
        if (pos == 1 || !head) {
```

```
            newNode->next = head;
```

```
            if (head) head->prev = newNode;
```

```
            head = newNode;
```

```
            if (!tail) tail = newNode;
```

```
            return;
```

```
        }
```

```
        DNode* temp = head;
```

```
        for (int i = 1; i < pos - 1 && temp->next; ++i)
```

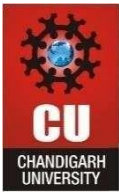
```
            temp = temp->next;
```

```
        newNode->next = temp->next;
```

```
        if (temp->next) temp->next->prev = newNode;
```

```
        newNode->prev = temp;
```

```
        temp->next = newNode;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        if (!newNode->next) tail = newNode;

    }

void remove(int pos) {

    if (!head) return;

    DNode* temp = head;

    if (pos == 1) {

        head = head->next;

        if (head) head->prev = NULL;

        else tail = NULL;

        delete temp;

        return;

    }

    for (int i = 1; i < pos && temp; ++i)

        temp = temp->next;

    if (!temp) return;

    if (temp->prev) temp->prev->next = temp->next;

    if (temp->next) temp->next->prev = temp->prev;

    if (temp == tail) tail = temp->prev;

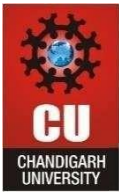
    delete temp;

}

void print() {

    DNode* temp = head;

    cout << "Doubly List: ";
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        while (temp) {

            cout << temp->data << " <-> ";

            temp = temp->next;

        }

        cout << "NULL\n";

    }

};

// ----- Circular Linked List -----

struct CNode {

    int data;

    CNode* next;

    CNode(int val) : data(val), next(NULL) {}

};

class CircularList {

    CNode* tail = NULL;

public:

    void insert(int pos, int val) {

        CNode* newNode = new CNode(val);

        if (!tail) {

            tail = newNode;

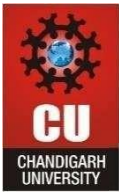
            tail->next = tail;

            return;

        }

    }

};
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
}

CNode* head = tail->next;

if (pos == 1) {

    newNode->next = head;

    tail->next = newNode;

    return;

}

CNode* temp = head;

for (int i = 1; i < pos - 1 && temp->next != head; ++i)

    temp = temp->next;

newNode->next = temp->next;

temp->next = newNode;

if (temp == tail) tail = newNode;

}


void remove(int pos) {

    if (!tail) return;

    CNode* head = tail->next;

    if (pos == 1) {

        if (head == tail) {

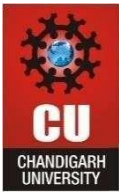
            delete head;

            tail = NULL;

            return;

        }

        tail->next = head->next;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        delete head;

        return;

    }

    CNode* temp = head;

    for (int i = 1; i < pos - 1 && temp->next != head; ++i)

        temp = temp->next;

    CNode* del = temp->next;

    if (del == head) return;

    temp->next = del->next;

    if (del == tail) tail = temp;

    delete del;

}

void print() {

    if (!tail) {

        cout << "Circular List: empty\n";

        return;

    }

    CNode* temp = tail->next;

    cout << "Circular List: ";

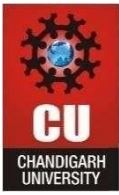
    do {

        cout << temp->data << " -> ";

        temp = temp->next;

    } while (temp != tail->next);

    cout << "(head)\n";
```

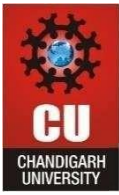



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
}  
  
};  
  
  
// ----- Main Program -----  
  
int main() {  
  
    SinglyList sl;  
  
    DoublyList dl;  
  
    CircularList cl;  
  
  
    int listType;  
  
    cout << "Choose List Type:\n1. Singly\n2. Doubly\n3. Circular\nEnter choice: ";  
  
    cin >> listType;  
  
  
  
    int ch, val, pos;  
  
    while (true) {  
  
        cout << "\n1. Insert\n2. Delete\n3. Display\n0. Exit\nEnter choice: ";  
  
        cin >> ch;  
  
        if (ch == 0) break;  
  
  
        switch (ch) {  
  
            case 1:  
  
                cout << "Enter value and position: ";  
  
                cin >> val >> pos;  
  
                if (listType == 1) sl.insert(pos, val);  
  
                else if (listType == 2) dl.insert(pos, val);  
  

```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
else cl.insert(pos, val);
```

```
break;
```

case 2:

```
cout << "Enter position to delete: ";
```

```
cin >> pos;
```

```
if (listType == 1) sl.remove(pos);
```

```
else if (listType == 2) dl.remove(pos);
```

```
else cl.remove(pos);
```

```
break;
```

case 3:

```
if (listType == 1) sl.print();
```

```
else if (listType == 2) dl.print();
```

```
else cl.print();
```

```
break;
```

default:

```
cout << "Invalid choice.\n";
```

```
}
```

```
}
```

```
return 0;
```

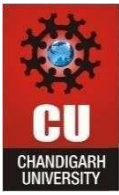
```
}
```

OUTPUT:

```
1 Choose List Type:
2 1. Singly
3 2. Doubly
4 3. Circular
5 Enter choice: 1
6
7 1. Insert
8 2. Delete
9 3. Display
0 0. Exit
1 Enter choice: 1
2 Enter value and position: 10 1
3
4 Enter choice: 1
5 Enter value and position: 20 2
6
7 Enter choice: 3
8 Singly List: 10 -> 20 -> NULL
9
0 Enter choice: 2
1 Enter position to delete: 1
2
3 Enter choice: 3
4 Singly List: 20 -> NULL
5 |
```

4.Learning Outcomes:

- Understood the structure and use-cases of singly, doubly, and circular linked lists.
- Learned insertion and deletion at beginning, end, and any position.
- Gained hands-on practice with pointer manipulation and dynamic memory.
- Handled edge cases like empty and single-node lists.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

5. Algorithm Analysis: