Experiment1.2

Student Name: Ankush **UID:** 23BCS12742

Branch: CSE **Section/Group:** KRG-3B

Semester: 5th **Date of Performance:**23/07/25

Subject Name: DAA(23CSH-301)

1) **Aim:** Write a program in to perform insertion and deletion operations on a singly, doubly and circular linked list. The program should allow inserting or deleting a node at the beginning, at the end, and at a specified position in the list. a given position.

C++ Code:

```
#include <iostream>
using namespace
std;  struct Node {
int data;
Node* next;
};
Node* head = NULL;
void display() {  Node*
temp = head;  while
(temp != NULL) {  cout
```

```
<< temp->data << " ->

";  temp = temp->next;

}

cout << "NULL\n";

}  void insertAtBeg(int val) {

Node* newNode = new

Node;  newNode->data = val;

newNode->next = head;  head

= newNode;

}  void insertAtEnd(int val) {

Node* newNode = new

Node;  newNode->data = val;

newNode->next = NULL;  if

(head == NULL) {  head =

newNode;

return;

}

Node* temp = head;  while

(temp->next != NULL)

temp = temp->next;  temp-

>next = newNode;

}
void insertAtPos(int val, int pos) {

if (pos == 1) {  insertAtBeg(val);

return;
```

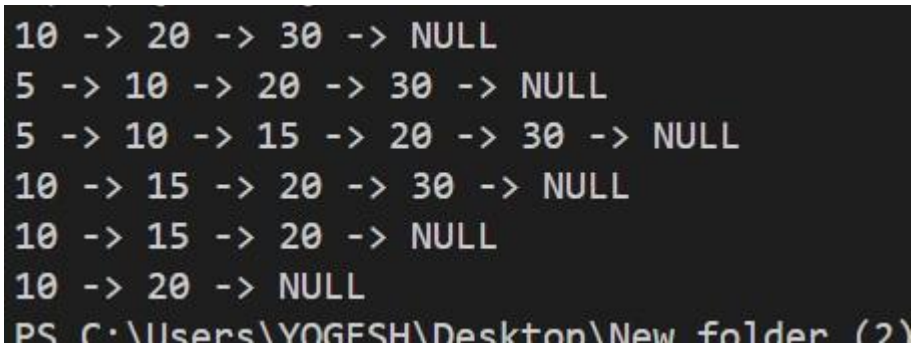
```
}
Node* newNode = new Node;  newNode->data
= val;  Node* temp = head;  for (int i = 1; i < pos
- 1 && temp != NULL; i++)  temp = temp-
>next;  if (temp == NULL) {  cout << "Invalid
position\n";  return;
}
newNode->next = temp->next;  temp-
>next = newNode;
}
void deleteFromBeg() {
if (head == NULL) {
cout << "List is empty\n";
return;
}
Node* temp = head;
head = head->next;
delete temp;

}
void deleteFromEnd() {
if (head == NULL) {
cout << "List is empty\n";
return;
}
```

```
if (head->next == NULL)
{  delete head;  head =
NULL;
return;
}
Node* temp = head;  while (temp-
>next->next != NULL)  temp =
temp->next;  delete temp->next;
temp->next = NULL;
}  void deleteFromPos(int
pos) {  if (head == NULL) {
cout << "List is empty\n";
return;  }  if (pos == 1) {
deleteFromBeg();  return;
}
Node* temp = head;  for (int i = 1; i < pos - 1 &&
temp->next != NULL; i++)  temp = temp->next;  if
(temp->next == NULL) {  cout << "Invalid position\n";
return;
}
Node* toDelete = temp->next;  temp-
>next = temp->next->next;  delete
toDelete;
```

```
}  int main() {
insertAtEnd(10);
insertAtEnd(20);
insertAtEnd(30);
display();
insertAtBeg(5);
display();
insertAtPos(15,
3);  display();
deleteFromBeg();
display();
deleteFromEnd();
display();
deleteFromPos(2);
display();  return
0;
}
```

Output:

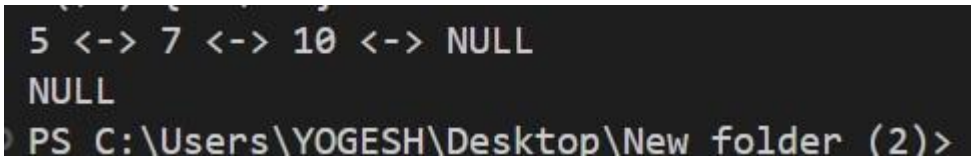


- **Doubly Linked List**

```
#include<iostream>
using namespace std;
struct Node {  int
data;  Node* prev;
Node* next;
};
Node* head = NULL;  void
insertAtBeginning(int val) {
Node* newNode = new Node;
newNode->data = val;
newNode->prev = NULL;
newNode->next = head;  if (head
!= NULL)  head->prev =
newNode;  head = newNode;
}  void insertAtEnd(int val) {
Node* newNode = new Node;
newNode->data = val;
newNode->next = NULL;
newNode->prev = NULL;  if
(head == NULL) {  head =
newNode;
return;
}
Node* temp = head;  while
(temp->next != NULL)
temp = temp->next;  temp-
>next = newNode;
newNode->prev = temp;
}  void insertAtPosition(int pos, int
val) {  if (pos == 0) {
insertAtBeginning(val);  return;
}
Node* temp = head;  for (int i = 0; temp !=
NULL && i < pos - 1; i++)  temp = temp->next;
if (temp == NULL) return;  Node* newNode =
new Node;  newNode->data = val;  newNode-
>next = temp->next;  newNode->prev = temp;  if
(temp->next != NULL)  temp->next->prev =
newNode;  temp->next = newNode;
}  void
deleteAtBeginning() {  if
(head == NULL) return;
Node* temp = head;  head
```

```
= head->next;  if (head !=
NULL)  head->prev =
NULL;  delete temp;
}  void deleteAtEnd() {  if
(head == NULL) return;  if
(head->next == NULL) {
delete head;
head = NULL;
return;
}
Node* temp = head;  while
(temp->next != NULL)  temp =
temp->next;  temp->prev->next
= NULL;  delete temp;  }  void
deleteAtPosition(int pos) {  if
(pos == 0) {
deleteAtBeginning();  return;
}
Node* temp = head;  for (int i = 0; temp !=
NULL && i < pos; i++)  temp = temp->next;
if (temp == NULL) return;  if (temp->prev !=
NULL)  temp->prev->next = temp->next;  if
(temp->next != NULL)  temp->next->prev =
temp->prev;  delete temp;  }  void display() {
Node* temp = head;  while (temp != NULL)
{  cout << temp->data << " <-> ";  temp =
temp->next;
}
cout << "NULL\n";
}  int main() {
insertAtEnd(10);
insertAtBeginning(5);
insertAtPosition(1, 7);
display();
deleteAtPosition(1);
deleteAtEnd();
deleteAtBeginning();
display();  return 0;
}
```

Output:



- **Circular Linked List:**

```
#include<iostream>
using namespace std;
struct Node {     int
data;
   Node* next;
};

Node* head = NULL;

void insertAtBeginning(int val) {
Node* newNode = new Node;
newNode->data = val;

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

   Node* temp = head;
while (temp->next != head)
temp = temp->next;

   newNode->next = head;
temp->next   =   newNode;
head = newNode;
}

void insertAtEnd(int val) {
Node* newNode = new Node;
newNode->data = val;

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

```
   Node* temp = head;
while (temp->next != head)
temp = temp->next;

   temp->next = newNode;
newNode->next = head;
}

void insertAtPosition(int pos, int val) {
if (pos == 0) {
insertAtBeginning(val);         return;
   }

   Node* temp = head;     for (int i = 0; i < pos - 1 &&
temp->next != head; i++)         temp = temp->next;

   Node* newNode = new Node;
newNode->data = val;
newNode->next = temp->next;
temp->next = newNode;
}

void deleteAtBeginning() {
if (head == NULL) return;

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

   Node* temp = head;
while (temp->next != head)
temp = temp->next;

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

void deleteAtEnd() {
   if (head == NULL) return;
```

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

   Node* temp = head;     while
(temp->next->next != head)
temp = temp->next;

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

void deleteAtPosition(int pos) {
if (pos == 0) {
deleteAtBeginning();        return;
   }

   Node* temp = head;     for (int i = 0; i < pos - 1 &&
temp->next != head; i++)        temp = temp->next;

   Node* toDelete = temp->next;
if (toDelete == head) return;

   temp->next = toDelete->next;
delete toDelete;
}

void display() {     if (head ==
NULL) {         cout << "List is
empty\n";        return;
   }

   Node* temp = head;     do {
cout << temp->data << " -> ";
temp = temp->next;     } while
(temp != head);     cout <<
"(head)\n";
}

int main() {
insertAtEnd(1);
insertAtBeginning(0);
insertAtEnd(2);
```

```
insertAtPosition(2, 5);
display();

   deleteAtPosition(2);
deleteAtEnd();
deleteAtBeginning();     display();

   return 0;
}
```

Output: