ASSIGNMENT

TASK NO: 8

SUBJECT: data structure and algorithm

SUBMITTED TO: madam hijab durrani

SUBMITTED BY: misbah ullah jan

CLASS CODE: bsse-2024A

SEMESTER: 3rd

DATE: 12/22/25

DEPARTMENT OF: software engineering

CECOS UNIVERSITY HAYATABAD PESHAWAR

LAB TASK 1:

1) Algorithm (Step by Step)

Algorithm: INSERT_AT_END(head, value)

1. New node N banao
2. N.data = value set karo
3. N.next = NULL set karo
4. Agar head == NULL ho
    - head = N
    - Stop
5. Warna temp = head rakho
6. Jab tak temp.next != NULL ho
    - temp = temp.next
7. temp.next = N
8. Stop

Algorithm: DISPLAY_LIST(head)

1. temp = head rakho
2. Jab tak temp != NULL ho
    - temp.data print karo
    - temp = temp.next
3. Stop

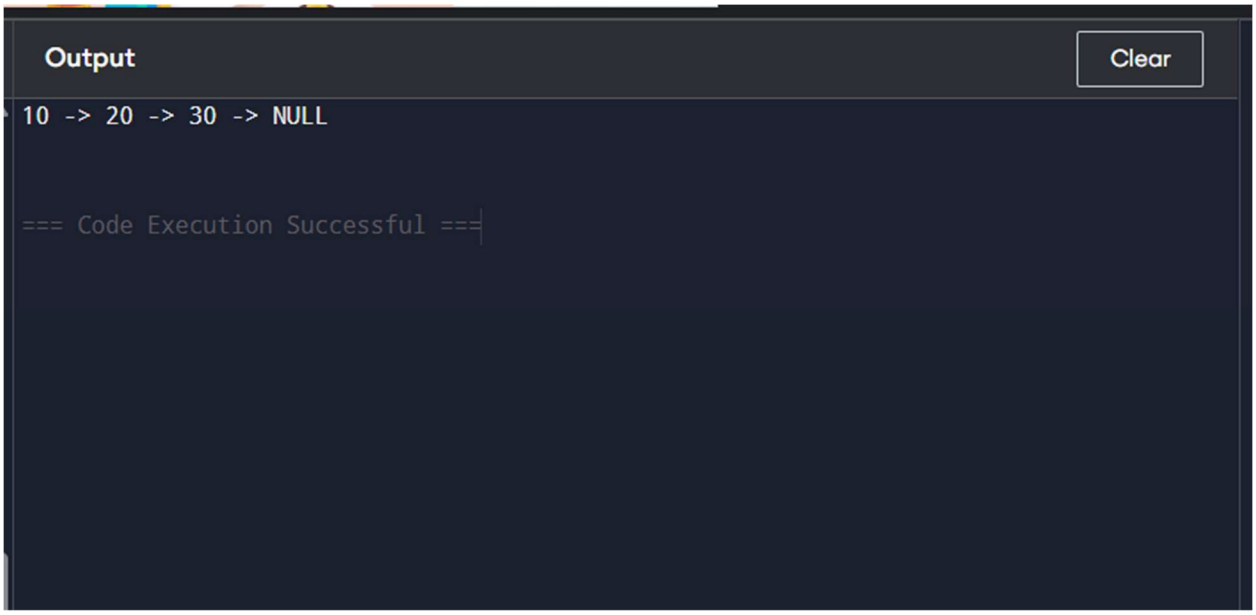
main.cpp

Share Run

```cpp
#include <iostream>
using namespace std;

// Node structure
struct Node {
    int data;
    Node* next;
};

Node* head = NULL;

// Insert at end
void insertAtEnd(int value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next = NULL;

    // If list is empty
    if (head == NULL) {
        head = newNode;
        return;
    }

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

```
29  }
30
31  // Display list
32  void displayList() {
33      Node* temp = head;
34      while (temp != NULL) {
35          cout << temp->data << " -> ";
36          temp = temp->next;
37      }
38      cout << "NULL" << endl;
39  }
40
41  int main() {
42      insertAtEnd(10);
43      insertAtEnd(20);
44      insertAtEnd(30);
45
46      displayList();
47
48      return 0;
49  }
50
www.programiz.com/html/online-compiler/
```

OUTPUT:



LAB TASK 2:

1) Algorithm

Algorithm: INSERT_AT_BEGINNING(head, value)

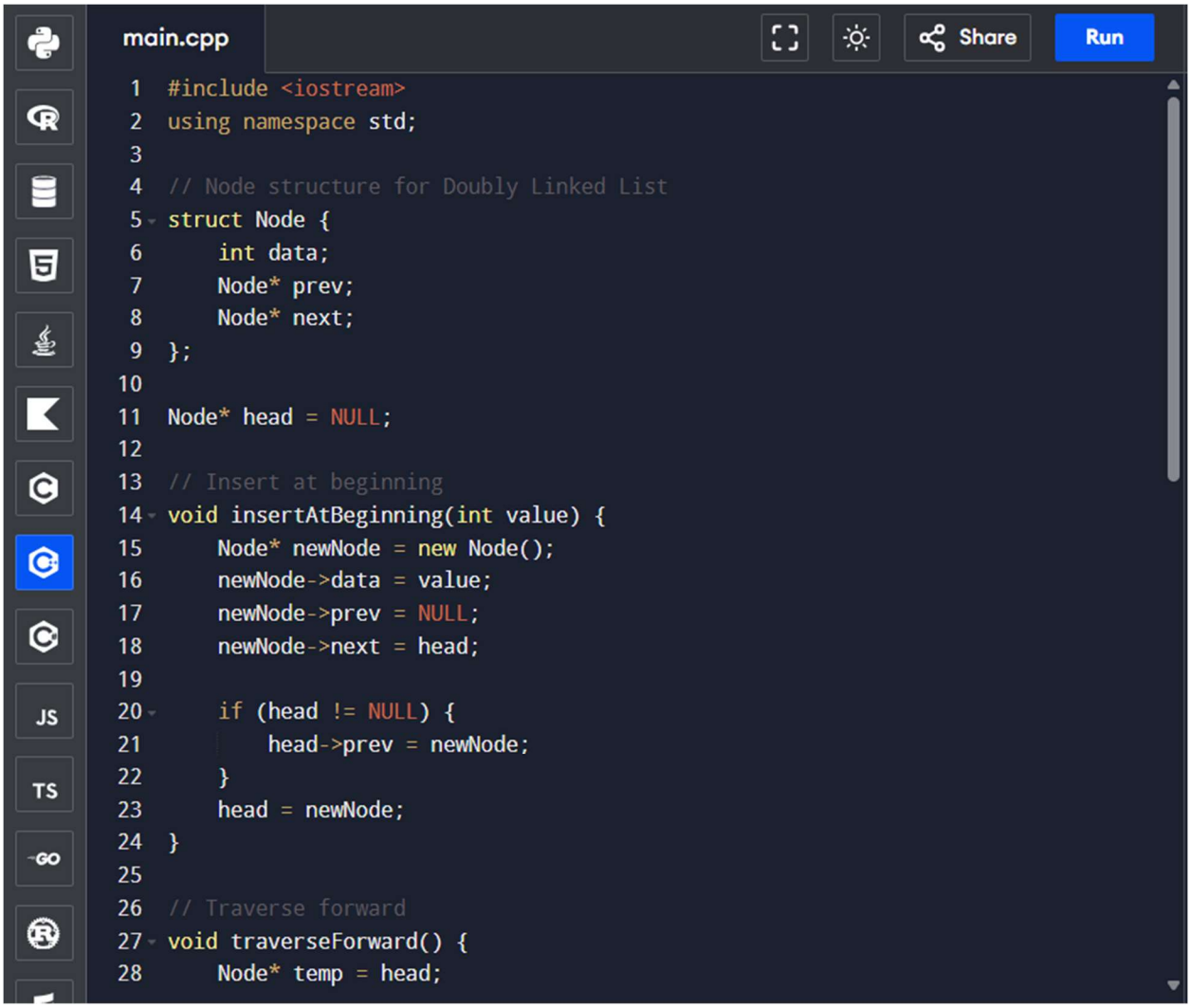
1. New node N banao
2. N.data = value set karo
3. N.prev = NULL set karo
4. N.next = head set karo
5. Agar head != NULL ho
    - head.prev = N
6. head = N update karo
7. Stop

Algorithm: TRAVERSE_FORWARD(head)

1. temp = head rakho
2. Jab tak temp != NULL ho
    - temp.data print karo
    - temp = temp.next
3. Stop

Algorithm: TRAVERSE_BACKWARD(head)

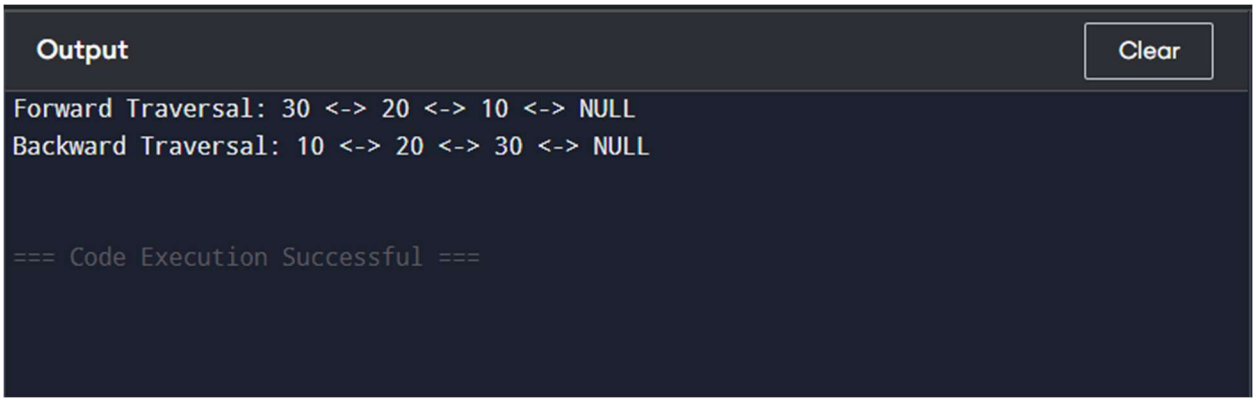
1. temp = head rakho
2. Jab tak temp.next != NULL ho
    - temp = temp.next
3. Ab jab tak temp != NULL ho
    - temp.data print karo
    - temp = temp.prev
4. Stop


main.cpp
Share
Run
#include <iostream>
using namespace std;
// Node structure for Doubly Linked List
struct Node {
int data;
Node* prev;
Node* next;
};
Node* head = NULL;
// Insert at beginning
void insertAtBeginning(int value) {
Node* newNode = new Node();
newNode->data = value;
newNode->prev = NULL;
newNode->next = head;
if (head != NULL) {
head->prev = newNode;
}
head = newNode;
}
// Traverse forward
void traverseForward() {
Node* temp = head;

```
    cout << "Forward Traversal: ";
    while (temp != NULL) {
        cout << temp->data << " <-> ";
        temp = temp->next;
    }
    cout << "NULL" << endl;
}

// Traverse backward
void traverseBackward() {
    Node* temp = head;

    // Move to last node
    while (temp != NULL && temp->next != NULL) {
        temp = temp->next;
    }

    cout << "Backward Traversal: ";
    while (temp != NULL) {
        cout << temp->data << " <-> ";
        temp = temp->prev;
    }
    cout << "NULL" << endl;
}

```

```
int main() {
    insertAtBeginning(10);
    insertAtBeginning(20);
    insertAtBeginning(30);

    traverseForward();
    traverseBackward();

    return 0;
}

```

OUTPUT:



LAB TASK 3:

1) Algorithm

Algorithm: INSERT_NODE_CIRCULAR(head, value)

1. New node N banao
2. N.data = value set karo
3. Agar head == NULL ho
    - head = N
    - N.next = head
    - Stop
4. Warna temp = head rakho
5. Jab tak temp.next != head ho
    - temp = temp.next
6. temp.next = N set karo
7. N.next = head set karo
8. Stop

main.cpp

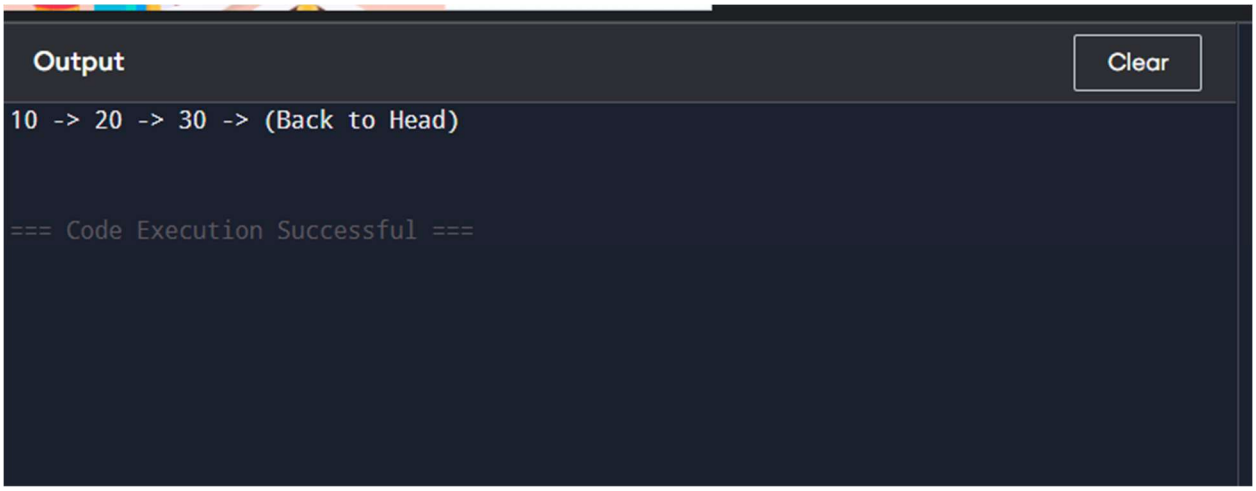
Share Run

```cpp
#include <iostream>
using namespace std;

// Node structure for Circular Linked List
struct Node {
    int data;
    Node* next;
};

Node* head = NULL;

// Insert node at end in Circular Linked List
void insertNode(int value) {
    Node* newNode = new Node();
    newNode->data = value;

    // If list is empty
    if (head == NULL) {
        head = newNode;
        newNode->next = head;
        return;
    }

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

```

```
29      temp->next = newNode;
30      newNode->next = head;
31  }
32
33  // Display Circular Linked List
34  void display() {
35      if (head == NULL) {
36          cout << "List is empty";
37          return;
38      }
39
40      Node* temp = head;
41      do {
42          cout << temp->data << " -> ";
43          temp = temp->next;
44      } while (temp != head);
45
46      cout << "(Back to Head)" << endl;
47  }
48
49  int main() {
50      insertNode(10);
51      insertNode(20);
52      insertNode(30);
53
```

```
54      display();
55
56      return 0;
57  }
58
```

OUTPUT:


Output
Clear
10 -> 20 -> 30 -> (Back to Head)
=== Code Execution Successful ===