

# DSA ASSIGNMENT - JUNE 24

Name – Vijay Kumar Saini

Roll.No - 23/11/EC/027

LeetCode Id - vijay\_kumar\_saini

GeekforGeeks - sainivij3yo0

Github Repo Link: <https://github.com/vijaysaini2613/jnu-ass>

1. Design a Singly Linked List and write functions to
  - Insert at beginning
  - Insert at end
  - Insert at any index
  - Delete at beginning
  - Delete at end

- Delete at index
- Delete first occurrence of node with given value
- Write at least 2 test cases for each operation

```
#include <iostream>
```

```
using namespace std;
```

```
// Node class to store data and a pointer to the next node
```

```
class Node {
```

```
public:
```

```
    int data;
```

```
    Node* next;
```

```
    Node(int val) {
```

```
        data = val;
```

```
        next = NULL;
```

```
    }
```

```
};
```

```
// Linked List class with basic operations
```

```
class SinglyLinkedList {
```

```
private:
```

```
    Node* head;
```

```
public:
```

```
    // Constructor: initially the list is empty
```

```
    SinglyLinkedList() {
```

```
        head = NULL;
```

```
    }
```

```
    // Insert a node at the beginning
```

```
    void insertAtBeginning(int val) {
```

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

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

```
        head = newNode;
```

```
    }
```

```
    // Insert a node at the end
```

```
    void insertAtEnd(int val) {
```

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

```
        if (head == NULL) {
```

```
            head = newNode;
```

```
            return;
```

```
        }
```

```
        Node* temp = head;
```

```
        while (temp->next != NULL) {
```

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

```
        }
```

```

    temp->next = newNode;
}

// Insert a node at a specific index
void insertAtIndex(int index, int val) {
    if (index == 0) {
        insertAtBeginning(val);
        return;
    }
    Node* newNode = new Node(val);
    Node* temp = head;
    for (int i = 0; i < index - 1 && temp != NULL; i++) {
        temp = temp->next;
    }
    if (temp == NULL) {
        cout << "Index out of range\n";
        return;
    }
    newNode->next = temp->next;
    temp->next = newNode;
}

```

```

// Delete the first node
void deleteAtBeginning() {
    if (head == NULL) {
        cout << "List is empty\n";
        return;
    }
    Node* temp = head;
    head = head->next;
    delete temp;
}

```

```

// Delete the last node
void deleteAtEnd() {
    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;
}

```

// Delete node at a specific index

```

void deleteAtIndex(int index) {
    if (head == NULL) {
        cout << "List is empty\n";
        return;
    }
    if (index == 0) {
        deleteAtBeginning();
        return;
    }
    Node* temp = head;
    for (int i = 0; i < index - 1 && temp != NULL; i++) {
        temp = temp->next;
    }
    if (temp == NULL || temp->next == NULL) {
        cout << "Index out of range\n";
        return;
    }
    Node* toDelete = temp->next;
    temp->next = toDelete->next;
    delete toDelete;
}

```

// Delete first node with given value

```

void deleteByValue(int val) {
    if (head == NULL) {
        cout << "List is empty\n";
        return;
    }
    if (head->data == val) {
        deleteAtBeginning();
        return;
    }
    Node* temp = head;
    while (temp->next != NULL && temp->next->data != val) {
        temp = temp->next;
    }
    if (temp->next == NULL) {
        cout << "Value not found\n";
        return;
    }
    Node* toDelete = temp->next;
    temp->next = toDelete->next;
    delete toDelete;
}

```

```

    }

    // Display the list
    void display() {
        Node* temp = head;
        while (temp != NULL) {
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
    }
};

```

```

int main() {
    SinglyLinkedList list;

    list.insertAtBeginning(10);
    list.insertAtBeginning(20);
    list.display(); // 20 10

    list.insertAtEnd(30);
    list.insertAtEnd(40);
    list.display(); // 20 10 30 40

    list.insertAtIndex(2, 25);
    list.display(); // 20 10 25 30 40

    list.deleteAtBeginning();
    list.display(); // 10 25 30 40

    list.deleteAtEnd();
    list.display(); // 10 25 30

    list.deleteAtIndex(1);
    list.display(); // 10 30

    list.deleteByValue(30);
    list.display(); // 10

    return 0;
}

```

## 2. Delete a Node in Singly Linked List

- Platform: GeeksforGeeks

- Link:

<https://www.geeksforgeeks.org/problems/delete-a-node-in-single-linked-list/1>

Output Window

Compilation Results

Custom Input

Y.O.G.I. (AI Bot)

Problem Solved Successfully

Test Cases Passed

2015 / 2015

Attempts : Correct / Total

1 / 1

Accuracy : 100%

Points Scored

2 / 2

Your Total Score: 15

Time Taken

0.46

Solve Next

Reverse a linked list

Frequency in a Linked List

```

1  /* Link list Node
2  struct Node
3  {
4      int data;
5      struct Node* next;
6  };
7  Node(int x){
8      data = x;
9      next = NULL;
10 }
11
12
13 class Solution {
14 public:
15     /* Function to delete a node from a linked list */
16     Node* deleteNode(Node* head, int x) {
17         // code here
18         if (x==1){
19             Node* temp=head->next;
20             delete head;
21             return temp;
22         }
23         head->next=deleteNode(head->next,x-1);
24         return head;
25     }
26 };

```

Custom Input

Compile & Run

Submit

### 3. Find Middle

- Platform: LeetCode
- Link: <https://leetcode.com/problems/middle-of-the-linked-list/description/>

All Submissions

Accepted 36 / 36 testcases passed

Sandhya\_005 submitted at Jun 24, 2025 23:03

Editorial

Solution

Runtime

0 ms | Beats 100.00%

Memory

10.05 MB | Beats 25.27%

Analyze Complexity

Code | C++

```

/**
 *
 *
 */

```

Testcase

Test Result

Accepted

Runtime: 0 ms

Case 1

Case 2

Input

head =

[1, 2, 3, 4, 5]

Output

[3, 4, 5]