```
#include <iostream>
using namespace std;
// Node Structure
struct Node
{
    int data;
    Node *next;
};
// Linked List Class
class LinkedList
{
private:
    Node *head;

public:
    // Constructor
    LinkedList()
    {
        head = NULL;
    }
    // Insert Function
    void insertAtBeginning(int value)
    {
        Node *newNode = new Node();
        newNode->data = value;
        newNode->next = head;
        head = newNode;
        cout << "Inserted " << value << " at the beginning." << endl;
    }
    // Insert at the End function
    void insertAtEnd(int value)
    {
        Node *newNode = new Node();
        newNode->data = value;
        newNode->next = NULL;
        if (head == NULL)
        {
            head = newNode;
            cout << "Inserted " << value << " as the first element." << endl;
            return;
        }
        Node *temp = head;
        while (temp->next != NULL)
    {
        temp = temp->next;
    }
    temp->next = newNode;
    cout << "Inserted " << value << " at the end." << endl;
}

// Insert at any Position
void insertAtPosition(int value, int position)
{
    if (position < 1)
    {
        cout << "Invalid position!" << endl;
        return;
    }
    if (position == 1)
    {
        insertAtBeginning(value);
        return;
    }

    Node *newNode = new Node();
    newNode->data = value;
    Node *temp = head;

    for (int i = 1; i < position - 1; i++)
    {
        if (temp == NULL)
        {
            cout << "Position out of bounds." << endl;
            return;
        }
        temp = temp->next;
    }
    if (temp == NULL)
    {
        cout << "Position out of bounds." << endl;
        return;
    }

    newNode->next = temp->next;
    temp->next = newNode;
    cout << "Inserted " << value << " at position " << position << "." << endl;
```

```
}

// Display Function
void display()
{
    if (head == NULL)
    {
        cout << "List is Empty." << endl;
        return;
    }
    Node *temp = head;
    cout << "Current List: ";
    while (temp != NULL)
    {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "NULL" << endl;
}

// Delete by value function
void deleteByValue(int value)
{
    if (head == NULL)
    {
        cout << "List is empty. Nothing to delete." << endl;
        return;
    }
    if (head->data == value)
    {
        Node *toDelete = head;
        head = head->next;
        delete toDelete;
        cout << "Deleted value " << value << "." << endl;
        return;
    }

    Node *temp = head;
    while (temp->next != NULL && temp->next->data != value)
    {
        temp = temp->next;
    }
    if (temp->next == NULL)
    {
        cout << "Value " << value << " not found in the list." << endl;
        return;
    }

    Node *toDelete = temp->next;
    temp->next = toDelete->next;
    delete toDelete;
    cout << "Deleted value " << value << "." << endl;
}

// Delete by position function
void deleteByPosition(int position)
{
    if (head == NULL)
    {
        cout << "List is empty." << endl;
        return;
    }
    if (position < 1)
    {
        cout << "Invalid position." << endl;
        return;
    }
    if (position == 1)
    {
        Node *toDelete = head;
        head = head->next;
        delete toDelete;
        cout << "Deleted node at position 1." << endl;
        return;
    }
    Node *temp = head;

    for (int i = 1; i < position - 1; i++)
    {
        if (temp == NULL || temp->next == NULL)
        {
            cout << "Position out of bounds." << endl;
            return;
        }
        temp = temp->next;
    }
```

```
        if (temp->next == NULL)
        {
            cout << "Position out of bounds." << endl;
            return;
        }

        Node *toDelete = temp->next;
        temp->next = toDelete->next;
        delete toDelete;
        cout << "Deleted node at position " << position << "." << endl;
    }
};
// Main Function
int main()
{
    LinkedList list;
    int choice, val, pos;

    do
    {
        cout << "\n--- DELETION MENU ---" << endl;
        list.display();
        cout << "1. Delete by Value" << endl;
        cout << "2. Delete by Position" << endl;
        cout << "3. Add Element (Append)" << endl;
        cout << "4. Exit" << endl;
        cout << "Choice: ";
        cin >> choice;

        switch (choice)
        {
        case 1:
            cout << "Enter value to delete: ";
            cin >> val;
            list.deleteByValue(val);
            break;
        case 2:
            cout << "Enter position to delete (1-based): ";
            cin >> pos;
            list.deleteByPosition(pos);
            break;
        case 3:
            cout << "Enter value to add: ";
            cin >> val;
            list.insertAtEnd(val);
            break;
        case 4:
            cout << "Exiting..." << endl;
            break;
        default:
            cout << "Invalid choice." << endl;
        }
    } while (choice != 4);

    return 0;
}
```