```c
#include <stdio.h>
#include <stdlib.h>

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

// Head structure
struct Head
{
    struct Node *start;
};

// Function to create a new node
struct Node *createNode(int data)
{
    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

// Function to insert a node at the end of the linked list
void insertAtEnd(struct Head *head, int data)
{
    struct Node *newNode = createNode(data);
    if (head->start == NULL)
    {
        head->start = newNode;
        return;
    }
    struct Node *temp = head->start;
    while (temp->next != NULL)
    {
        temp = temp->next;
    }
    temp->next = newNode;
    newNode->prev = temp;
}

// Function to insert a node at the beginning of the linked list
void insertAtBeginning(struct Head *head, int data)
{
```

```
    struct Node *newNode = createNode(data);
    if (head->start == NULL)
    {
        head->start = newNode;
        return;
    }
    newNode->next = head->start;
    head->start->prev = newNode;
    head->start = newNode;
}

// Function to insert a node after the nth node of the linked list
void insertAfterNthNode(struct Head *head, int data, int n)
{
    struct Node *newNode = createNode(data);
    if (head->start == NULL)
    {
        head->start = newNode;
        return;
    }
    struct Node *temp = head->start;
    for (int i = 1; i < n && temp != NULL; i++)
    {
        temp = temp->next;
    }
    if (temp == NULL)
    {
        printf("Invalid position\n");
        return;
    }
    newNode->next = temp->next;
    newNode->prev = temp;
    if (temp->next != NULL)
    {
        temp->next->prev = newNode;
    }
    temp->next = newNode;
}

// Function to delete a node from the linked list
void deleteNode(struct Head *head, int data)
{
    struct Node *temp = head->start;
    while (temp != NULL && temp->data != data)
    {
        temp = temp->next;
    }
    if (temp == NULL)
```

```
    {
        printf("Node not found\n");
        return;
    }
    if (temp->prev != NULL)
    {
        temp->prev->next = temp->next;
    }
    else
    {
        head->start = temp->next;
    }
    if (temp->next != NULL)
    {
        temp->next->prev = temp->prev;
    }
    free(temp);
}

// Function to display the linked list
void display(struct Head *head)
{
    struct Node *temp = head->start;
    printf("Linked list: ");
    while (temp != NULL)
    {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

// Main function
int main()
{
    struct Head head;
    head.start = NULL;
    int choice, data, n;
    while (1)
    {
        printf("1. Insert at end\n");
        printf("2. Insert at beginning\n");
        printf("3. Insert after nth node\n");
        printf("4. Delete a node\n");
        printf("5. Display\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```
        switch (choice)
        {
        case 1:
            printf("Enter data: ");
            scanf("%d", &data);
            insertAtEnd(&head, data);
            break;
        case 2:
            printf("Enter data: ");
            scanf("%d", &data);
            insertAtBeginning(&head, data);
            break;
        case 3:
            printf("Enter data: ");
            scanf("%d", &data);
            printf("Enter position: ");
            scanf("%d", &n);
            insertAfterNthNode(&head, data, n);
            break;
        case 4:
            printf("Enter data: ");
            scanf("%d", &data);
            deleteNode(&head, data);
            break;
        case 5:
            display(&head);
            break;
        case 6:
            exit(0);
        default:
            printf("Invalid choice\n");
        }
    }
    return 0;
}
```

```
/tmp/X9HA4PzEw2.o
1. Insert at end
2. Insert at beginning
3. Insert after nth node
4. Delete a node
5. Display
6. Exit
Enter your choice: 1
Enter data: 10
1. Insert at end
2. Insert at beginning
3. Insert after nth node
4. Delete a node
5. Display
6. Exit
Enter your choice: 1
Enter data: 20
1. Insert at end
2. Insert at beginning
3. Insert after nth node
4. Delete a node
5. Display
6. Exit
Enter your choice: 1
Enter data: 30
```

```
Enter your choice: 3
Enter data: 40
Enter position: 2
1. Insert at end
2. Insert at beginning
3. Insert after nth node
4. Delete a node
5. Display
6. Exit
Enter your choice: 5
Linked list: 10 20 40 30
1. Insert at end
2. Insert at beginning
3. Insert after nth node
4. Delete a node
5. Display
6. Exit
Enter your choice: 4
Enter data: 40
1. Insert at end
2. Insert at beginning
3. Insert after nth node
4. Delete a node
5. Display
6. Exit
```

```
Enter your choice: 2
Enter data: 40
1. Insert at end
2. Insert at beginning
3. Insert after nth node
4. Delete a node
5. Display
6. Exit
Enter your choice: 5
Linked list: 40 10 20 30
1. Insert at end
2. Insert at beginning
3. Insert after nth node
4. Delete a node
5. Display
6. Exit
Enter your choice: 6
```