01/02/24

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node *prev;
    struct Node *next;
};
struct Node *createNode(int value)
{
    struct Node *newNode = (struct Node *)malloc
                           (sizeof(struct Node));
    if (newNode == NULL)
    {
        fprintf("memory allocation failed\n");
        exit(1);
    }
    newNode->data = value;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}
void insertToLeft(struct Node **head, int value)
{
    struct Node *newNode = createNode(value);
    if (*head == NULL){
        *head = newNode;
    }
    else
    {
        newNode->next = *head;
        (*head)->prev = newNode;
        *head = newNode;
    }
}
```

```
void deleteNode(struct Node **head, int value)
{
  if (*head == NULL)
  {
    printf("List is empty\n");
    return;
  }
  struct Node *current = *head;
  while (current != NULL)
  {
    if (current->data == value)
    {
      if (current->prev != NULL){
        current->prev->next = current->next;
      } else {
        *head = current->next;
      }
      if (current->next != NULL) {
        current->next->prev = current->prev;
      }
      free(current);
      printf("Node with value %d deleted.\n", value);
      return;
    }
    current = current->next;
  }
  printf("Node with value %d not found\n", value);
}
void displayList(struct Node *head)
{
  if (head == NULL)
  {
    printf("List is empty\n");
    return;
  }
```

```
printf("Doubly Linked list:");
while(head != NULL)
{
    printf("%d <-> ", head->data);
    head = head->next;
}
printf("NULL\n");
}
int main()
{
    struct node *head = NULL;
    int choice, value;
    do {
        printf("\n1. Insert to left 2. Delete by value
               3. Display 4. Exit\n");
        printf("Enter your choice");
        scanf("%d", &choice);
        switch(choice)
        {
            case 1: printf("Enter the value to insert:");
                    scanf("%d", &value);
                    insertToLeft(&head, value);
                    break;
            case 2: printf("Enter the value to delete:");
                    scanf("%d", &value);
                    deleteNode(&head, value);
                    break;
            case 3: displayList(head);
                    break;
            case 4: printf("Exiting the program\n");
                    break;
            default: printf("Invalid choice. Please enter
                     a valid option\n");
        }
    } while(choice != 4);
```

Leetcode: Linked list

Output:

1. Insert to left
2. Delete by value
3. Display
4. Exit

Enter your choice: 1
Enter the value to insert: 12

Enter your choice: 1
Enter the value to insert: 13

Enter your choice: 1
Enter value to insert: 14

Enter your choice: 1
Enter value to insert: 5

Enter your choice: 3
Doubly linked list: 12 <=> 13 <=> 14 <=> NULL

Enter your choice: 2
Enter the value: 13
Node with value 13 deleted

Enter your choice: 3
Doubly linked list: 12 <=> 14 <=> NULL

leetcode : Reversing linked list

```
struct ListNode* reverseBetween(struct ListNode* head,
int left, int right) {
if (head == NULL || left == right)
{
    return head;
}
struct ListNode *dummy = (struct ListNode*)
malloc(sizeof(struct ListNode));
dummy->next = head;
struct ListNode *prev = dummy;
for(int i=1; i<left; ++i)
    prev = prev->next;
}
struct ListNode* current = prev->next;
struct ListNode* next = NULL;
struct ListNode *tail = current;

for(int i=left; i<=right; i++)
{
    struct ListNode *temp = current->next;
    current->next = next;
    next = current;
    current = temp;
}
prev->next = next;
tail->next = current;

struct ListNode *result = dummy->next;
free(dummy);
return result;
```