

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

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

struct Node* createList(struct Node* head, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;
    if (head == NULL) {
        return newNode;
    }
    struct Node* temp = head;
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = newNode;
    newNode->prev = temp;
    return head;
}

struct Node* insertAtBeginning(struct Node* head, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = head;
    if (head != NULL)
        head->prev = newNode;
    head = newNode;
    return head;
}

struct Node* DeleteNode(struct Node* head, int value) {
    struct Node* temp = head;
    while (temp != NULL && temp->data != value)
        temp = temp->next;
    if (temp == NULL)
        printf("Node with value %d not found\n", value);
    else {
        if (temp == head) {
            printf("Head node deleted.\n");
            head = temp->next;
        } else {
            temp->prev->next = temp->next;
            if (temp->next != NULL)
                temp->next->prev = temp->prev;
            free(temp);
            printf("Node %d deleted.\n", value);
        }
    }
    return head;
}

void display(struct Node* head) {
    struct Node* temp = head;
    if (head == NULL)
        printf("List is empty.\n");
    else {
        printf("Doubly Linked List: ");
        while (temp != NULL) {
            printf("%d <-> ", temp->data);
            temp = temp->next;
        }
        printf("NULL\n");
    }
}

int main() {
    struct Node* head = NULL;
    int choice, value, key;
    while (1) {
        printf("\n--- MENU ---\n");
        printf("1. Create Node\n");
        printf("2. Insert at Beginning\n");
        printf("3. Delete Node by Value\n");
        printf("4. Display List\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
    }
}
```

```
52     struct Node* temp = head;
53
54     while (temp != NULL && temp->data != value)
55         temp = temp->next;
56
57     if (temp == NULL) {
58         printf("Node with value %d not found\n", value);
59         return head;
60     }
61
62     if (temp->prev != NULL)
63         temp->prev->next = temp->next;
64     else
65         head = temp->next;
66
67     if (temp->next != NULL)
68         temp->next->prev = temp->prev;
69
70     free(temp);
71     printf("Node %d deleted.\n", value);
72
73     return head;
74 }
75
76 void display(struct Node* head) {
77     struct Node* temp = head;
78
79     if (head == NULL)
80         printf("List is empty.\n");
81     else
82     {
83         printf("Doubly Linked List: ");
84         while (temp != NULL)
85             printf("%d <-> ", temp->data);
86         temp = temp->next;
87     }
88     printf("NULL\n");
89 }
90
91 int main() {
92     struct Node* head = NULL;
93     int choice, value, key;
94
95     while (1)
96     {
97         printf("\n--- MENU ---\n");
98         printf("1. Create Node\n");
99         printf("2. Insert at Beginning\n");
100        printf("3. Delete Node by Value\n");
101        printf("4. Display List\n");
102        printf("5. Exit\n");
103        printf("Enter your choice: ");
104    }
105}
```

The screenshot shows the Code::Blocks IDE interface with the file 'doublylinkedlist.c' open. The code implements a doubly linked list with functions for creating nodes, inserting at beginning, deleting by value, displaying the list, and exiting. The code uses standard C libraries and includes comments explaining the logic.

```
Start here X doublylinkedlist.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran Tools Tools+ Plugins DocBlocks Settings Help
90     printf("NULL\n");
91
92
93
94     int main() {
95         struct Node* head = NULL;
96         int choice, value, key;
97
98         while (1) {
99             printf("... MENU ... \n");
100            printf("1. Create Node\n");
101            printf("2. Insert at beginning\n");
102            printf("3. Delete Node by Value\n");
103            printf("4. Display List\n");
104            printf("5. Exit\n");
105            printf("Enter choice: ");
106            scanf("%d", &choice);
107
108            switch (choice) {
109                case 1:
110                    printf("Enter value to insert: ");
111                    scanf("%d", &value);
112                    head = createList(head, value);
113                    break;
114
115                case 2:
116                    printf("Enter value to insert at beginning: ");
117                    scanf("%d", &value);
118                    head = insertAtBeginning(head, value);
119                    break;
120
121                case 3:
122                    printf("Enter value to delete: ");
123                    scanf("%d", &value);
124                    head = deleteNode(head, value);
125                    break;
126
127                case 4:
128                    display(head);
129                    break;
130
131                case 5:
132                    exit(0);
133
134                default:
135                    printf("Invalid choice!\n");
136            }
137        }
138
139        return 0;
140    }
141
142 }
```

The screenshot shows a Windows command-line interface window titled 'D:\chethan01\doublylinkedlist'. It displays the execution of the program 'doublylinkedlist.exe'. The user interacts with the program by entering menu choices and data values, such as creating nodes with values 233 and 566, inserting at the beginning, and deleting node 233.

```
D:\chethan01\doublylinkedlist.exe
... MENU ...
1. Create Node
2. Insert at beginning
3. Delete Node by Value
4. Display List
5. Exit
Enter choice: 1
Enter value to insert: 233

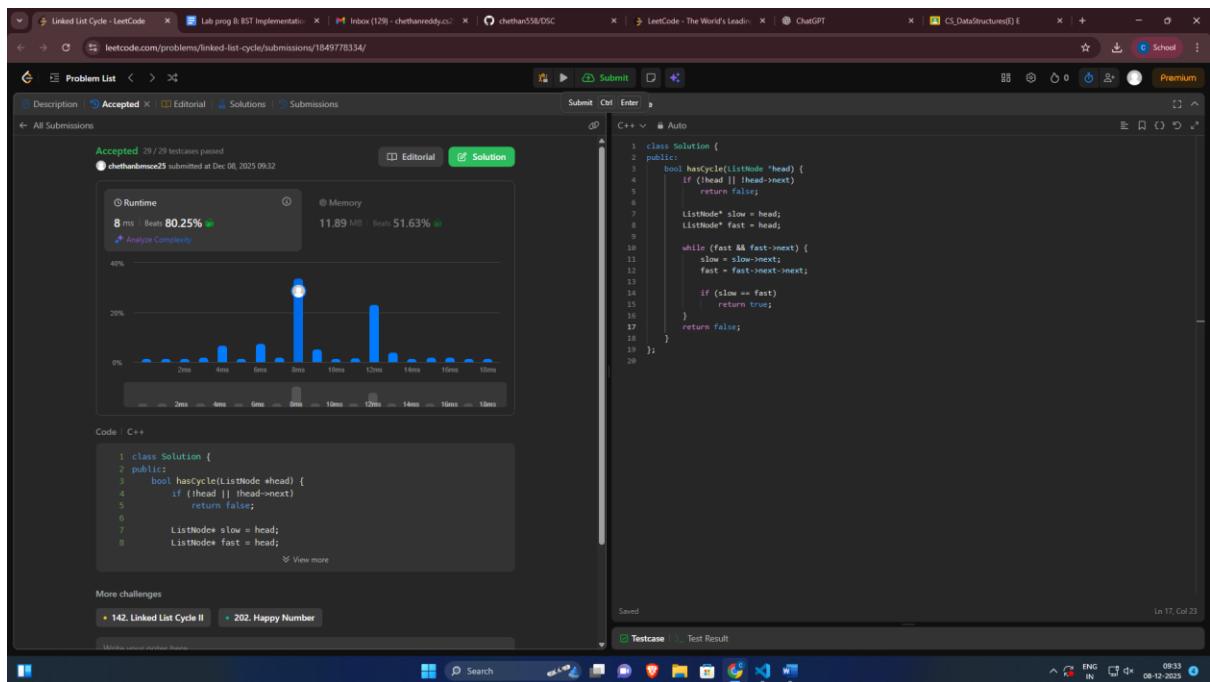
... MENU ...
1. Create Node
2. Insert at beginning
3. Delete Node by Value
4. Display List
5. Exit
Enter choice: 1
Enter value to insert: 566

... MENU ...
1. Create Node
2. Insert at beginning
3. Delete Node by Value
4. Display List
5. Exit
Enter choice: 2
Enter value to insert at beginning: 1222

... MENU ...
1. Create Node
2. Insert at beginning
3. Delete Node by Value
4. Display List
5. Exit
Enter choice: 3
Enter value to delete: 233
Node 233 deleted.

... MENU ...
1. Create Node
2. Insert at beginning
3. Delete Node by Value
4. Display List
5. Exit
Enter choice: 4
doubly Linked List: 1222 <-> 566 <-> NULL

... MENU ...
1. Create Node
2. Insert at beginning
3. Delete Node by Value
4. Display List
5. Exit
Enter choice:
```



3) Implementation of double linked list

pseudocode:

Function (creatnode (int)):

create a node = (newnode)

newnode->data = data

newnode->prev = newnode->next = NULL

if (tail = NULL) head = tail = newnode;

else

tail->next = newnode;

newnode->prev = tail;

tail = newnode;

end

Function insert at beginning (int data)

if head is null

head = tail = newnode;

else

head->prev = newnode;

~~head~~ Newnode->next = head;

head = newnode

end if

Function insert at end (int data):

if (tail == NULL)

tail->next = newnode;

newnode->prev = tail;

tail = newnode;

end if

Function ~~delete~~ (int val),

temp = head

while (temp != NULL & temp.data != val)

temp = temp->next;

Shot on OnePlus

chE Reddy

#include <stdiops>
 #include <stdlib.h>
 struct Node {
 int data;
 struct Node* prev;
 struct Node* next;
 };
 struct Node* createlist (struct Node* head, int data)
 {
 struct Node* newnode = (struct Node*) malloc (sizeof (struct Node));
 newnode->data = data;
 newnode->prev = NULL;
 newnode->next = NULL;
 if (head == NULL) {
 return newnode;
 }
 struct Node* temp = head;
 while (temp->next != NULL) {
 temp = temp->next;
 }
 temp->next = newnode;
 newnode->prev = temp;
 return head;
 }

~~struct Node* insertatBeginning (struct Node* head, int data)~~
 struct Node* newnode = (struct Node*) malloc (sizeof (struct Node));
 newnode->data = data;
 newnode->prev = NULL;
 newnode->next = head;
 if (head != NULL)
 head->prev = newnode;
 newnode->next = head;
 return head;

Shot on OnePlus

chE_Reddy

DATE: _____

```

struct node* deleteNode (struct node* head, int value) {
    struct node* temp = head;
    while (temp != NULL && temp->data != value)
        temp = temp->next;
    if (temp == NULL)
        printf("Node with value %d not found.\n", value);
    else
        head = temp->next;
    if (temp->next == NULL)
        temp->next = temp->prev;
    free(temp);
    printf("Node %d deleted.\n", value);
    return head;
}

```

```

void display (struct node* head) {
    struct node* temp = head;
    if (head == NULL)
        printf("List is empty.\n");
    else
        {
}

```

```

        printf("Doubly linked list:");
        while (temp != NULL) {
            printf(" | d->", temp->data);
            temp = temp->next;
}

```

printf("NULL\n");



Shot on OnePlus

chE_Reddy

DATE: _____
 PAGE: _____

```

int main() {
  struct node * head = NULL;
  int choice, value, key;
  while (1) {
    printf("1. --- Menu ---\n");
    printf("2. Create Node\n");
    printf("3. Insert at beginning\n");
    printf("4. Delete node by value\n");
    printf("5. Display list\n");
    printf("6. EXIT\n");
    printf("Enter choice: ");
    scanf("%d", &choice);
    switch (choice) {
      case 1:
        printf("Enter value to insert: ");
        scanf("%d", &value);
        head = createList(head, value);
        break;
      case 2:
        printf("Enter value to insert at beginning: ");
        scanf("%d", &value);
        head = insertatbeginning(head, value);
        break;
      case 3:
        printf("Enter value to delete: ");
        scanf("%d", &value);
        head = deleteNode(head, value);
        break;
      case 4:
        display(head);
        break;
    }
  }
}
  
```

Output:

-- menu--

1. Create Node
2. Insert at beginning
3. Delete Node by value
4. Display List
5. Exit

Enter choice: 2

Enter value to insert: 23

Enter choice: 1

Enter value to insert: 45

Enter choice: 4

Doubly Linked List: 23->45->NULL

Enter choice: 3

Enter value to delete: 45

Node 45 deleted

Enter your choice: 1

Doubly Linked List: 23->NULL

Enter choice: 3

Enter value to delete: 23

Node with value 23 not found!

Enter your: 1

Enter value to insert: 555

Enter your choice: 4

Doubly linked list 23->555->NULL



Shot on OnePlus

chE_Reddy

8) To construct a binary search tree and traverse the tree using inorder, preorder, postorder and display elements in the tree.

leekode is linked list cycle

class solution {

public:

```
bool hascycle(ListNode * head) {
    if (!head || !head->next)
        return false;
    ListNode * slow = head;
    ListNode * fast = head;
    while (fast && fast->next)
        slow = slow->next;
        fast = fast->next->next;
    if (slow == fast)
        return true;
}
```

return false;

?
? *OR*
status



Shot on OnePlus

chE_Reddy