

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node {
5     int data;
6     struct Node *prev;
7     struct Node *next;
8 };
9
10 struct Node *head = NULL;
11
12 struct Node* createNode(int value) {
13     struct Node *newNode = (struct Node*) malloc(sizeof(struct Node));
14     newNode->data = value;
15     newNode->prev = NULL;
16     newNode->next = NULL;
17     return newNode;
18 }
19
20 void createList(int value) {
21     struct Node *newNode = createNode(value);
22
23     if (head == NULL) {
24         head = newNode;
25         return;
26     }
27
28     struct Node *temp = head;
29     while (temp->next != NULL)
30         temp = temp->next;
31
32     temp->next = newNode;
33     newNode->prev = temp;
34 }
35
36 void insertLeft(int target, int value) {
37     if (head == NULL) {
38         printf("List is empty.\n");
39         return;
40     }
41
42     struct Node *temp = head;
43
44     while (temp != NULL && temp->data != target)
45         temp = temp->next;
46
47     if (temp == NULL) {
48         printf("Value %d not found in list.\n", target);
49         return;
50 }
```

```
13     while (temp != NULL && temp->data != target) {
14         temp = temp->next;
15     }
16     if (temp == NULL) {
17         printf("Value %d not found in list.\n", target);
18         return;
19     }
20     struct Node *newNode = createNode(value);
21
22     if (temp == head) {
23         newNode->next = head;
24         head->prev = newNode;
25         head = newNode;
26     } else {
27         newNode->prev = temp->prev;
28         newNode->next = temp;
29         temp->prev->next = newNode;
30         temp->prev = newNode;
31     }
32     printf("Inserted %d to the left of %d\n", value, target);
33 }
34
35 void deleteValue(int value) {
36     if (head == NULL) {
37         printf("List is empty.\n");
38         return;
39     }
40
41     struct Node *temp = head;
42     while (temp != NULL && temp->data != value)
43         temp = temp->next;
44
45     if (temp == NULL) {
46         printf("Value %d not found.\n", value);
47         return;
48     }
49
50     if (temp == head) {
51         head = head->next;
52         if (head != NULL)
53             head->prev = NULL;
54     } else if (temp->next == NULL) {
55         temp->prev->next = NULL;
56     } else {
57         temp->prev->next = temp->next;
58         temp->next->prev = temp->prev;
59     }
60 }
```

```
    if (temp == head) {
        head = head->next;
        if (head != NULL)
            head->prev = NULL;
    } else if (temp->next == NULL) {
        temp->prev->next = temp->next;
        temp->next->prev = temp->prev;
    } else {
        temp->prev->next = temp->next;
        temp->next->prev = temp->prev;
    }

    free(temp);
    printf("Deleted %d\n", value);
}

void display() {
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }

    struct Node *temp = head;
    printf("List: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    createList(10);
    createList(20);
    createList(40);
    printf("Initial List:\n");
    display();
    insertLeft(40, 25);
    display();
    deleteValue(20);
    display();
    return 0;
}
```

C:\Users\Admin\Desktop\1BF24CS183\doublyll.exe

```
Initial List:
List: 10 20 40
Inserted 25 to the left of 40
List: 10 20 25 40
Deleted 20
List: 10 25 40

Process returned 0 (0x0)  execution time : 0.000 s
Press any key to continue.
```