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

// node creation
struct Node {
  int data;
  struct Node* next;
  struct Node* prev;
};

// insert node at the front
void insertFront(struct Node* head, int data) {
  struct Node* ptr = (struct Node*)malloc(sizeof(struct Node));
  ptr->data = data;

  ptr->next = head;
  ptr->prev = NULL;

  if (head != NULL)
    head->prev =ptr;
    head= ptr;
}

// insert a node after a specific node
void insertAfter(struct Node* head,struct Node* prev_node, int dat
  if (prev_node == NULL) {
    printf("previous node cannot be null");
    return;
  }

  struct Node* ptr = (struct Node*)malloc(sizeof(struct Node));
  ptr->data = data;

  ptr->next = prev_node->next;
  prev_node->next = ptr;
  ptr->prev = prev_node;

  if (ptr->next != NULL)
    ptr->next->prev =ptr;
```

```
40: }
41:
42: // insert a newNode at the end of the list
43: void insertEnd(struct Node* head, int data){
44:   struct Node* ptr = (struct Node*)malloc(sizeof(struct Node));
45:   ptr->data = data;
46:   struct Node* p= head;
47:
48:   ptr->next = NULL;
49:
50:   if (head == NULL) {
51:     ptr->prev = NULL;
52:     head = ptr;
53:     return;
54:   }
55:
56:   while (p->next != NULL)
57:     p = p->next;
58:     p->next = ptr;
59:     ptr->prev =p;
60: }
61:
62: // delete a node from the doubly linked list
63: void deleteNode(struct Node* head, struct Node* del_node) {
64:
65:   if (head == NULL || del_node == NULL)
66:     return;
67:
68:   if (head == del_node)
69:      head = del_node->next;
70:
71:   if (del_node->next != NULL)
72:     del_node->next->prev = del_node->prev;
73:
74:   if (del_node->prev != NULL)
75:     del_node->prev->next = del_node->next;
76:
77:   free(del_node);
78: }
```

```c

// print the doubly linked list
void displayList(struct Node* node) {
  struct Node* last;

  while (node != NULL) {
    printf("%d->", node->data);
    last = node;
    node = node->next;
  }
  if (node == NULL)
    printf("NULL\n");
}

int main() {


  /* Initialize nodes */
struct Node *head=NULL;
struct Node *one = NULL;
struct Node *two = NULL;
struct Node *three = NULL;

/* Allocate memory */
one = malloc(sizeof(struct Node));
two = malloc(sizeof(struct Node));
three = malloc(sizeof(struct Node));

/* Assign data values */
one->data = 1;
two->data = 2;
three->data = 3;

/* Connect nodes */
one->next = two;
one->prev = NULL;

two->next = three;
two->prev = one;
```

```
118:
119: three->next = NULL;
120: three->prev = two;
121:
122: /* Save address of first node in head */
123: head = one;
124:
125:   insertEnd(&head, 5);
126:   insertFront(&head, 1);
127:   insertFront(&head, 6);
128:   insertEnd(&head, 9);
129:
130:   // insert 11 after head
131:   insertAfter(head,two,11);
132:
133:   // insert 15 after the seond node
134:   insertAfter(head,head->next,15);
135:
136:   displayList(head);
137:
138:   // delete the last node
139:   deleteNode(&head, head->next->next->next->next->next);
140:
141:   displayList(head);
142:   return 0;
143: }
```