```c
#include <stdio.h>

#include <stdlib.h>

typedef struct Node {

   int data;

   struct Node* next;

} Node;

Node* createNode(int data) {

   Node* newNode = (Node*)malloc(sizeof(Node));

   if (newNode == NULL) {

      printf("Memory allocation failed\n");

      exit(1);

   }

   newNode->data = data;

   newNode->next = NULL;

   return newNode;

}

void printList(Node* head) {

   Node* temp = head;

   while (temp != NULL) {

      printf("%d -> ", temp->data);

      temp = temp->next;

   }

   printf("NULL\n");

}

void insertEnd(Node** head, int data) {

   Node* newNode = createNode(data);
```

```
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    Node* temp = *head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}
Node* findMiddle(Node* head) {
    if (head == NULL) {
        return NULL;
    }
    Node *slow = head, *fast = head;
    while (fast != NULL && fast->next != NULL) {
        slow = slow->next;
        fast = fast->next->next;
    }
    return slow;
}
void insertAfterMiddle(Node** head, int data) {
    Node* middle = findMiddle(*head);
    if (middle == NULL) {
        printf("List is empty\n");
        return;
    }
    Node* newNode = createNode(data);
    newNode->next = middle->next;
    middle->next = newNode;
}
void deleteMiddle(Node** head) {
    if (*head == NULL) {
        printf("List is empty\n");
        return;
    }
    Node* middle = findMiddle(*head);
    if (middle == NULL) {
        return;
    }
    if (middle == *head) {
        Node* temp = *head;
        *head = (*head)->next;
        free(temp);
        return;
    }
    Node* prev = NULL;
    Node* temp = *head;
    while (temp != middle) {
        prev = temp;
        temp = temp->next;
    }
    prev->next = middle->next;
    free(middle);
}
int main() {
    Node* head = NULL;
    insertEnd(&head, 1);
    insertEnd(&head, 2);
    insertEnd(&head, 3);
    insertEnd(&head, 4);
    insertEnd(&head, 5);

    printf("Original List:\n");
    printList(head);
    insertAfterMiddle(&head, 10);
    printf("After inserting 10 after the middle node:\n");
    printList(head);
    deleteMiddle(&head);
    printf("After deleting the middle node:\n");
    printList(head);

    return 0;
}
```