Linked List Singly Using Strructure In C

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
   int data;
   struct Node* next;
};
struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   if (!newNode) {
      printf("Memory allocation error\n");
      exit(1);
   }
   newNode->data = data;
   newNode->next = NULL;
   return newNode;
}
void insertAtBeginning(struct Node** head, int data) {
   struct Node* newNode = createNode(data);
   newNode->next = *head;
   *head = newNode;
   printf("Inserted %d at the beginning\n", data);
}
void insertAtEnd(struct Node** head, int data) {
   struct Node* newNode = createNode(data);
   if (*head == NULL) {
      *head = newNode;
   } else {
      struct Node* temp = *head;
      while (temp->next != NULL) {
```

```
            temp = temp->next;
        }
        temp->next = newNode;
    }
    printf("Inserted %d at the end\n", data);
}
void deleteNode(struct Node** head, int key) {
    struct Node* temp = *head;
    struct Node* prev = NULL;
    if (temp != NULL && temp->data == key) {
        *head = temp->next;
        free(temp);
        printf("Deleted %d\n", key);
        return;
    }
    while (temp != NULL && temp->data != key) {
        prev = temp;
        temp = temp->next;
    }
    if (temp == NULL) return;
    prev->next = temp->next;
    free(temp);
    printf("Deleted %d\n", key);
}
void printList(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
```

```
}
int main() {
    struct Node* head = NULL;
    insertAtEnd(&head, 8);
    printList(head);

    insertAtEnd(&head, 2);
    printList(head);

    insertAtEnd(&head, 7);
    printList(head);

    insertAtBeginning(&head, 5);
    printList(head);

    deleteNode(&head, 8);
    printList(head);
    return 0;
}
```

Output:

```
Inserted 8 at the end
8 -> NULL
Inserted 2 at the end
8 -> 2 -> NULL
Inserted 7 at the end
8 -> 2 -> 7 -> NULL
Inserted 5 at the beginning
5 -> 8 -> 2 -> 7 -> NULL
Deleted 8
5 -> 2 -> 7 -> NULL
```

Linked List Double

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
   int data;
   struct Node* next;
   struct Node* prev;
};
struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   if (!newNode) {
      printf("Memory allocation error\n");
      exit(1);
   }
   newNode->data = data;
   newNode->next = NULL;
   newNode->prev = NULL;
   return newNode;
}
void insertAtBeginning(struct Node** head, int data) {
   struct Node* newNode = createNode(data);
   newNode->next = *head;
   if (*head != NULL) {
      (*head)->prev = newNode;
   }
   *head = newNode;
   printf("Inserted %d at the beginning\n", data);
}
void insertAtEnd(struct Node** head, int data) {
   struct Node* newNode = createNode(data);
```

```
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    struct Node* temp = *head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
    newNode->prev = temp;
    printf("Inserted %d at the end\n", data);
}
void deleteNode(struct Node** head, int key) {
    struct Node* temp = *head;
    while (temp != NULL && temp->data != key) {
        temp = temp->next;
    }
    if (temp == NULL) return;
    if (temp == *head) {
        *head = temp->next;
    }
    if (temp->next != NULL) {
        temp->next->prev = temp->prev;
    }
    if (temp->prev != NULL) {
        temp->prev->next = temp->next;
    }
    free(temp);
    printf("Deleted %d\n", key);
}
void printListForward(struct Node* head) {
```

```
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d <-> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
void printListBackward(struct Node* head) {
    struct Node* temp = head;
    if (temp == NULL) return;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    while (temp != NULL) {
        printf("%d <-> ", temp->data);
        temp = temp->prev;
    }
    printf("NULL\n");
}
int main() {
    struct Node* head = NULL;

    insertAtEnd(&head, 1);
    insertAtEnd(&head, 2);
    insertAtEnd(&head, 3);

    printListForward(head);

    insertAtBeginning(&head, 0);
    printListForward(head);
    printListForward(head);
```

```
    deleteNode(&head, 2);
    printListForward(head);

    printListBackward(head);
    return 0;
}
```

Output:

```
Inserted 2 at the end
Inserted 3 at the end
1 <-> 2 <-> 3 <-> NULL
Inserted 0 at the beginning
0 <-> 1 <-> 2 <-> 3 <-> NULL
3 <-> 2 <-> 1 <-> 0 <-> NULL
Deleted 2
0 <-> 1 <-> 3 <-> NULL
3 <-> 1 <-> 0 <-> NULL
```

Linked List Circular

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (!newNode) {
        printf("Memory allocation error\n");
```

```
        exit(1);
    }
    newNode->data = data;
    newNode->next = newNode; // Point to itself initially
    return newNode;
}
void insertAtBeginning(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
    } else {
        struct Node* temp = *head;
        while (temp->next != *head) {
            temp = temp->next;
        }
        newNode->next = *head;
        temp->next = newNode;
        *head = newNode;
    }
    printf("Inserted %d at the beginning\n", data);
}
void insertAtEnd(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
    } else {
        struct Node* temp = *head;
        while (temp->next != *head) {
            temp = temp->next;
        }
```

```
        temp->next = newNode;
        newNode->next = *head;
    }
    printf("Inserted %d at the end\n", data);
}
void deleteNode(struct Node** head, int key) {
    if (*head == NULL) return;
    struct Node* temp = *head;
    struct Node* prev = NULL;
    do {
        if (temp->data == key) break;
        prev = temp;
        temp = temp->next;
    } while (temp != *head);
    if (temp->data != key) return;
    if (temp->next == temp) {
        free(temp);
        *head = NULL;
        printf("Deleted %d\n", key);
        return;
    }
    if (temp == *head) {
        prev = *head;
        while (prev->next != *head) {
            prev = prev->next;
        }
        *head = temp->next;
        prev->next = *head;
    } else {
        prev->next = temp->next;
```

```
    }
    free(temp);
    printf("Deleted %d\n", key);
}
void printList(struct Node* head) {
    if (head == NULL) return;
    struct Node* temp = head;
    do {
        printf("%d -> ", temp->data);
        temp = temp->next;
    } while (temp != head);
    printf("(head)\n");
}
int main() {
    struct Node* head = NULL;

    insertAtEnd(&head, 1);
    insertAtEnd(&head, 2);
    insertAtEnd(&head, 3);
    printList(head);

    insertAtBeginning(&head, 0);
    printList(head);

    deleteNode(&head, 2);
    printList(head);

    deleteNode(&head, 1);
    printList(head);
```

```
    deleteNode(&head, 0);
    printList(head);

    deleteNode(&head, 3);
    printList(head);
    return 0;
}
```

Output:

```
Inserted 1 at the end
Inserted 2 at the end
Inserted 3 at the end
1 -> 2 -> 3 -> (head)
Inserted 0 at the beginning
0 -> 1 -> 2 -> 3 -> (head)
Deleted 2
0 -> 1 -> 3 -> (head)
Deleted 1
0 -> 3 -> (head)
Deleted 0
3 -> (head)
Deleted 3
```