Assignment 2 - Data Structures

Question: Implementation of Linked List Operations in C

The provided code implements several linked list operations in C, including insertion at the beginning, insertion at the end, insertion after a key, deletion from the beginning, deletion from the end, deletion with a key, and display functionality. The question is implicit; it asks us to examine the correctness and completeness of the C code provided.

Algorithm and C Program

The code provided in the image is analyzed, and since a specific question hasn't been asked, the provided code that was in the image has been embedded into HTML and provided as a C program. Each function in the code performs a specific operation on a singly linked list.

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

struct node{
    int data;
    struct node *link;
};

//Insert at the first
void insertFirst(struct node*h)
{
    struct node *newNode;
    int ele;
    newNode =(struct node *)malloc(sizeof(struct node *));
    if(newNode==NULL)
        printf("Insuffucient Memory\n");
    else
    {
        printf("Enter the element: ");
        scanf("%d",&ele);
        newNode->data=ele;
        newNode->link=h->link;
        h->link=newNode;
    }
}

//Insert at the last
void insertLast(struct node *h)
{
    struct node *newNode,*ptr=h;
    int ele;
```

```
    newNode =(struct node *)malloc(sizeof(struct node *));
    if(newNode==NULL)
        printf("Insuffucient Memory\n");
    else
    {
        printf("Enter the element: ");
        scanf("%d",&ele);
        newNode->data=ele;
        newNode->link=NULL;
        while (ptr->link!=NULL)
        {
            ptr=ptr->link;
        }
        ptr->link=newNode;
    }
}

//Insert the element after keyvalue
//IF keyvalue is not present, Inserting at last
void insertatKey(struct node *h)
{
    struct node *newNode,*ptr=h;
    int ele, keyvalue;
    newNode =(struct node *)malloc(sizeof(struct node *));
    if(newNode==NULL)
        printf("Insuffucient Memory\n");
    else
    {
        printf("Enter the element: ");
        scanf("%d",&ele);
        newNode->data=ele;
        printf("Enter the keyvalue after which element to add:- ");
        scanf("%d",&keyvalue);
        while (ptr->link!=NULL && ptr->data!=keyvalue)
        {
            ptr=ptr->link;
        }
        newNode->link=ptr->link;
        ptr->link=newNode;
    }
}

//Delete the element from the beginning
int deleteFirst(struct node*h)
{
    struct node *ptr;
    int temp;
    if(h->link==NULL)
        printf("Empty linked list\n");
    else
    {
        ptr=h->link;
        h->link=ptr->link;
```

```
        temp=ptr->data;
        free(ptr);
        return temp;
    }
    return -1; // Return -1 if the list is empty
}

//Delete the elemnet from the last
int deleteLast(struct node *h)
{
    struct node *ptr1,*ptr2;
    int temp;
    if(h->link==NULL)
        printf("Empty linked list\n");
    else
    {
        ptr1=h->link;
        ptr2=h;
        while (ptr1->link!=NULL)
        {
            ptr2=ptr1;
            ptr1=ptr1->link;
        }
        ptr2->link=NULL;
        temp=ptr1->data;
        free(ptr1);
        return temp;
    }
    return -1; // Return -1 if the list is empty
}

//For deletion with the keyvalue
int deletewithKey(struct node *h)
{
    struct node *ptr1,*ptr2;
    int temp, keyvalue;
    if(h->link==NULL)
        printf("Empty linked list\n");
    else
    {
        printf("Enter the keyvalue for which element to delete:- \n");
        scanf("%d",&keyvalue);
        ptr1=h->link;
        ptr2=h;
        while (ptr1->link!=NULL && ptr1->data!=keyvalue)
        {
            ptr2=ptr1;
            ptr1=ptr1->link;
        }
        if (ptr1->data==keyvalue)
        {
            temp=ptr1->data;
            ptr2->link=ptr1->link;
```

```
            free(ptr1);
            return temp;
        }
        else
        {
            printf("No keyvalue found\n");
            return -1; // Return -1 if keyvalue is not found
        }
    }
    return -1; // Return -1 if the list is empty
}

void display(struct node *h)
{
    struct node *ptr;
    ptr=h;
    printf("Elements are: ");
    while (ptr->link!=NULL)
    {
        ptr=ptr->link;
        printf("%d ",ptr->data);
    }
    printf("\n");
}

int main()
{
    printf("Singly Linked List\n");
    struct node *Head;
    Head=(struct node*)malloc(sizeof(struct node *));
    if(Head==NULL)
        printf("Insuffiecient memor\n");
    else
    {
        Head->link=NULL;
    }
    deleteFirst(Head);
    insertFirst(Head);
    insertFirst(Head);
    display(Head);
    insertLast(Head);
    insertLast(Head);
    display(Head);
    insertatKey(Head);
    insertatKey(Head);
    display(Head);
    deleteFirst(Head);
    deleteFirst(Head);
    display(Head);
    deleteLast(Head);
    deleteLast(Head);
    display(Head);
    deletewithKey(Head);
```

```
    display(Head);
    deletewithKey(Head);
    display(Head);
    return 0;
}
```