Implementation of Single Linked List

Write a C program to implement the following operations on Singly Linked List.

(i) Insert a node in the beginning of a list.

(ii) Insert a node after P

(iii) Insert a node at the end of a list

(iv) Find an element in a list

(v) FindNext

(vi) FindPrevious

(vii) isLast

(viii) isEmpty

(ix) Delete a node in the beginning of a list.

(x) Delete a node after P

(xi) Delete a node at the end of a list

(xii) Delete the List

PROGRAM:

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

// Define the structure for a node
struct Node {
   int data;
   struct Node* next;
};

// Function to create a new node
struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->next = NULL;
   return newNode;
}

// Function to insert a node at the beginning
void insertAtBeginning(struct Node** head, int data) {
   struct Node* newNode = createNode(data);
```

```
    newNode->next = *head;
    *head = newNode;
}

// Function to insert a node after a given node
void insertAfterNode(struct Node* prevNode, int data) {
    if (prevNode == NULL) {
        printf("The given previous node cannot be NULL\n");
        return;
    }
    struct Node* newNode = createNode(data);
    newNode->next = prevNode->next;
    prevNode->next = newNode;
}

// Function to insert a node at the end
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;
}

// Function to find an element in the list
struct Node* findElement(struct Node* head, int data) {
    struct Node* current = head;
    while (current != NULL) {
        if (current->data == data) {
            return current;
        }
```

```c
        current = current->next;
    }
    return NULL;
}

// Function to find the next node of a given node
struct Node* findNext(struct Node* node) {
    if (node == NULL) {
        return NULL;
    }
    return node->next;
}

// Function to find the previous node of a given node
struct Node* findPrevious(struct Node* head, struct Node* node) {
    if (head == NULL || head == node) {
        return NULL;
    }
    struct Node* current = head;
    while (current->next != NULL && current->next != node) {
        current = current->next;
    }
    return (current->next == node) ? current : NULL;
}

// Function to check if a node is the last node
int isLast(struct Node* node) {
    return (node != NULL && node->next == NULL);
}

// Function to check if the list is empty
int isEmpty(struct Node* head) {
    return (head == NULL);
}

// Function to delete the node at the beginning
```

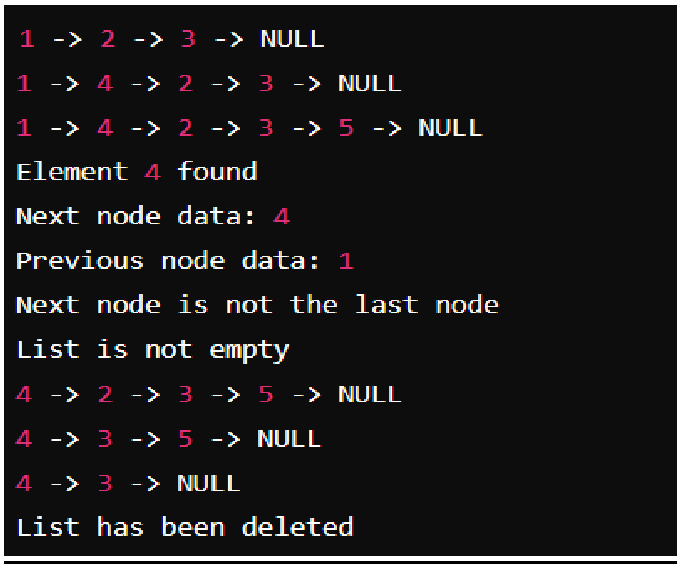
```
void deleteAtBeginning(struct Node** head) {
    if (*head == NULL) {
        return;
    }
    struct Node* temp = *head;
    *head = (*head)->next;
    free(temp);
}

// Function to delete the node after a given node
void deleteAfterNode(struct Node* prevNode) {
    if (prevNode == NULL || prevNode->next == NULL) {
        return;
    }
    struct Node* temp = prevNode->next;
    prevNode->next = temp->next;
    free(temp);
}

// Function to delete the node at the end
void deleteAtEnd(struct Node** head) {
    if (*head == NULL) {
        return;
    }
    if ((*head)->next == NULL) {
        free(*head);
        *head = NULL;
        return;
    }
    struct Node* temp = *head;
    while (temp->next->next != NULL) {
        temp = temp->next;
    }
    free(temp->next);
    temp->next = NULL;
}
```

```
// Function to delete the entire list
void deleteList(struct Node** head) {
   struct Node* current = *head;
   struct Node* nextNode;
   while (current != NULL) {
      nextNode = current->next;
      free(current);
      current = nextNode;
   }
   *head = NULL;
}

// Function to print the list
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;

   // Insert nodes at the beginning
   insertAtBeginning(&head, 3);
   insertAtBeginning(&head, 2);
   insertAtBeginning(&head, 1);
   printList(head);

   // Insert node after the first node
   insertAfterNode(head, 4);
   printList(head);
```

```
// Insert node at the end
insertAtEnd(&head, 5);
printList(head);

// Find an element in the list
struct Node* foundNode = findElement(head, 4);
if (foundNode) {
   printf("Element 4 found\n");
} else {
   printf("Element 4 not found\n");
}

// Find the next node of the head
struct Node* nextNode = findNext(head);
if (nextNode) {
   printf("Next node data: %d\n", nextNode->data);
}

// Find the previous node of node containing 4
struct Node* prevNode = findPrevious(head, foundNode);
if (prevNode) {
   printf("Previous node data: %d\n", prevNode->data);
}

// Check if a node is the last node
if (isLast(nextNode)) {
   printf("Next node is the last node\n");
} else {
   printf("Next node is not the last node\n");
}

// Check if the list is empty
if (isEmpty(head)) {
   printf("List is empty\n");
} else {
   printf("List is not empty\n");
```

```
    }

    // Delete node at the beginning
    deleteAtBeginning(&head);
    printList(head);

    // Delete node after the first node
    deleteAfterNode(head);
    printList(head);

    // Delete node at the end
    deleteAtEnd(&head);
    printList(head);

    // Delete the entire list
    deleteList(&head);
    if (isEmpty(head)) {
        printf("List has been deleted\n");
    }

    return 0;
}
```

OUTPUT:



```
1 -> 2 -> 3 -> NULL
1 -> 4 -> 2 -> 3 -> NULL
1 -> 4 -> 2 -> 3 -> 5 -> NULL
Element 4 found
Next node data: 4
Previous node data: 1
Next node is not the last node
List is not empty
4 -> 2 -> 3 -> 5 -> NULL
4 -> 3 -> 5 -> NULL
4 -> 3 -> NULL
List has been deleted
```