Experiments\SinglyLinkedLIst.c

```c
#include<stdio.h>
#include<malloc.h>
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 new element at the beginning of the singly linked list
void insertAtFirst(struct Node* *start, int data) {
 struct Node* newNode = createNode(data);
 newNode->next = *start;
 *start = newNode;
}
// Function to insert a new element at the end of the singly linked list
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 insert a new element at a specific position in the singly linked list
void insertAtPosition(struct Node** head, int data, int position) {
 struct Node* newNode = createNode(data);
 if (position == 0) {
 insertAtFirst(head,data);
 return;
 }
 struct Node* temp = *head;
 for (int i = 0; temp != NULL && i < position - 1; i++) {
    temp = temp->next;
 }
 if (temp == NULL) {
 printf("Position out of range\n");
 free(newNode);
 return;
```

```c
 }
 newNode->next = temp->next;
 temp->next = newNode;
}
// Function to delete the first node of the singly linked list
void deleteFromFirst(struct Node** head) {
 if (*head == NULL) {
 printf("List is empty\n");
 return;
 }
 struct Node* temp = *head;
 *head = temp->next;
 free(temp);
}
// Function to delete the last node of the singly linked list
void deleteFromEnd(struct Node** head) {
 if (*head == NULL) {
 printf("List is empty\n");
 return;
 }
 struct Node* temp = *head;
 if (temp->next == NULL) {
 free(temp);
 *head = NULL;
 return;
 }
 while (temp->next->next != NULL) {
 temp = temp->next;
 }
 free(temp->next);
 temp->next = NULL;
}
// Function to delete a node at a specific position in the singly linked list
void deleteAtPosition(struct Node** head, int position) {
 if (*head == NULL) {
 printf("List is empty\n");
 return;
 }
 struct Node* temp = *head;
 if (position == 0) {
 deleteFromFirst(head);
 return;
 }
 for (int i = 0; temp != NULL && i < position - 1; i++) {
 temp = temp->next;
 }
 if (temp == NULL || temp->next == NULL) {
 printf("Position out of range\n");
 return;
 }
```

```c
 struct Node* next = temp->next->next;
 free(temp->next);
 temp->next = next;
}
// Function to print the LinkedList
void print(struct Node* head) {
 struct Node* temp = head;
 while (temp != NULL) {
 printf("%d -> ", temp->data);
 temp = temp->next;
 }
 printf("NULL\n");
}
// Driver Code
int main() {
 struct Node* head = NULL;

 insertAtFirst(&head, 10);
 printf("Linked list after inserting the node:10 at the beginning \n");
 print(head);

 printf("Linked list after inserting the node:20 at the end \n");
 insertAtEnd(&head, 20);
 print(head);

 printf("Linked list after inserting the node:5 at the end \n");
 insertAtEnd(&head, 5);
 print(head);

 printf("Linked list after inserting the node:30 at the end \n");
 insertAtEnd(&head, 30);
 print(head);

 printf("Linked list after inserting the node:15 at position 2 \n");
 insertAtPosition(&head, 15, 2);
 print(head);

 printf("Linked list after deleting the first node: \n");
 deleteFromFirst(&head);
 print(head);

 printf("Linked list after deleting the last node: \n");
 deleteFromEnd(&head);
 print(head);

 printf("Linked list after deleting the node at position 1: \n");
 deleteAtPosition(&head, 1);
 print(head);
 return 0;
}
```