

WAP for insertion and deletion for single linked list

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

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

```
struct Node {  
    int data;  
    struct Node* next;  
};
```

```
void insertAtBeginning(struct Node** head, int value) {  
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));  
    newNode->data = value;  
    newNode->next = *head;  
    *head = newNode;  
}
```

```
void insertAtEnd(struct Node** head, int value) {  
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));  
    struct Node* temp = *head;  
    newNode->data = value;  
    newNode->next = NULL;  
  
    if (*head == NULL) {  
        *head = newNode;  
        return;  
    }
```

```
}
```

```
while (temp->next != NULL) {
```

```
    temp = temp->next;
```

```
}
```

```
temp->next = newNode;
```

```
}
```

```
void insertAtPosition(struct Node** head, int value, int position) {
```

```
    if (position <= 0) {
```

```
        printf("Invalid position\n");
```

```
        return;
```

```
    }
```

```
    if (position == 1 || *head == NULL) {
```

```
        insertAtBeginning(head, value);
```

```
        return;
```

```
    }
```

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

```
    newNode->data = value;
```

```
    struct Node* temp = *head;
```

```
    int count = 1;
```

```
    while (count < position - 1 && temp->next != NULL) {
```

```

    temp = temp->next;

    count++;

}

if (count < position - 1) {

    printf("Invalid position\n");

    return;

}

newNode->next = temp->next;

temp->next = newNode;

}

void deleteAtBegining(struct Node** head){

    if (*head == NULL) {

        printf("The linkedlist is already empty\n");

        return;

    }

    else{

        struct Node* first = *head;

        *head = (*head)->next;

        free(first);

    }

}

void deleteAtEnd(struct Node** head){

    if(*head==NULL) {

```

```

    printf("The linkedlist is already empty\n");

    return;
}

else{

    struct Node* temp = *head;

    while(temp->next->next!=NULL){

        temp = temp->next;

    }

    struct Node* lastNode = temp->next;

    temp->next=NULL;

    free(lastNode);

}

}

void deleteAtIndex(struct Node **head, int pos) {

    if(*head == NULL){

        printf("The Linked List is Empty \n");

    }

    else{

        struct Node* temp = *head;

        pos--;

        while(pos-- && temp!=NULL){

            temp = temp->next;

        }

        if(temp==NULL){

            printf("pos not exist\n");

        }

    }

}

```

```

else{

    struct Node* nxt = temp->next->next;

    struct Node* del = temp->next;

    temp->next = temp->next->next;

    free(del);

}

}

}

void displayLinkedList(struct Node* head) {

    struct Node* temp = head;

    if (temp == NULL) {

        printf("Linked list is empty.\n");

        return;

    }

    while (temp != NULL) {

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

        temp = temp->next;

    }

    printf("NULL\n");

}

int main() {

    struct Node* head = NULL;

```

```
insertAtBeginning(&head, 1);  
insertAtBeginning(&head, 2);  
insertAtBeginning(&head, 3);  
  
printf("Linked list after insertion at the beginning: ");  
displayLinkedList(head);  
  
insertAtEnd(&head, 5);  
insertAtEnd(&head, 6);  
  
printf("Linked list after insertion at the end: ");  
displayLinkedList(head);  
  
insertAtPosition(&head, 7, 2);  
insertAtPosition(&head, 8, 4);  
  
printf("Linked list after insertion at specific positions: ");  
displayLinkedList(head);  
  
printf("deletion\n");  
deleteAtBeginning(&head);  
deleteAtIndex(&head,1);  
deleteAtEnd(&head);  
displayLinkedList(head);
```

```
return 0;
```

```
}
```

OUTPUT:

```
Linked list after insertion at the beginning: 3 -> 2 -> 1 -> NULL
Linked list after insertion at the end: 3 -> 2 -> 1 -> 5 -> 6 -> NULL
Linked list after insertion at specific positions: 3 -> 7 -> 2 -> 8 -> 1 -> 5 -> 6 -> NULL
deletion
7 -> 8 -> 1 -> 5 -> NULL

Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```