8/1/2024

WEEK-5

Single_linked_list_deletion and display

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

struct Node {
    int data;
    struct Node* next;
};
    printf("NULL\n");
}
struct Node * insertAtFront(struct Node* head,
int data) {
    struct Node * newNode = (struct Node*)
    malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = head;
    return newNode;
}
struct Node * deleteAtFront(struct Node *head) {
    if (head == NULL) {
        printf("List is empty. Cannot delete.\n");
        return head;
    }
```

```
    struct Node * temp = head;
    head = head -> next;
    free(temp);
    return head;
}
struct Node * deleteAtLast(struct Node* head){
    if (head == NULL) {
        printf("List is empty. Cannot delete.\n");
        return head;
    }
    if (head -> next == NULL) {
        free(head);
        return NULL;
    }
    struct Node * current = head;
    struct Node * prev = NULL;

    while (current = current -> next);
    }
    free (current);
    prev -> next = NULL;
    return head;
}
struct Node * deleteAtPosition(struct Node * head,
    int position) {
```

```
if (head == NULL) {
    printf("Invalid position. Position should be >= 1.\n");
    return head;
}
struct Node * current = head;
struct Node * prev = NULL;

if (position == 1) {
    head = head -> next;
    free(current)
    return head;
}
for (int i = 1; i < position && current != NULL; i++) {
    prev = current;
    current = current -> next;
}


int main() {
    struct Node * head = NULL;
    int choice, value, position;
    do {
        printf("\n1. Insert at front \n2. Insert at
            front \n3. Delete at last 4. Delete at
            position \n 6. Display list \n 5. exit");
```

```
printf ("Enter your choice:");
scanf ("%d", &choice);

switch (choice) {
    case 1:
        printf ("Enter value to insert in front:");
        scanf ("%d", &value);
        head = insertAtFront (head);
        display list (head);
        break;
    case 2:
        head = deleteAtFront (head);
        display list (head);
        break;
    case 3:
        head = deleteAtFront (head);
        display list (head)
        break;
    case 4:
        printf ("Enter position:");
        scanf ("%d", &position);
        head = deleteAtposition (head, position);
    case 5:
        display list (head);
    default:
        printf ("invalid choice");
```

```
    }
} while (choice != 0);
  struct Node * current = head;
  while (current != NULL) {
    struct Node * temp = current;
    current = current -> next;
    free (temp);
  }
  return 0
}
```

Output :-

Single linked list deletion and display :-

1. Insert at front
2. Delete at front
3. Delete at last
4. Display ~~at~~ list
5. Delete the position

Enter your choice : 1
Enter the value to insert at front : 87
87 -> NULL

Enter your choice : 1

Enter the value to insert at front : 93
93 -> 87 -> NULL

Enter the value to insert at front : 27
27 -> 93 -> 87 -> NULL

Enter your choice : 2
93 -> 87 -> NULL

Enter your choice : 3
93 -> NULL

Enter your choice : 4
Enter the position to delete : 1
NULL.

18/11/24