LAB-04 22/01/2024

1. Single Linked List Operations:

```
#include <stdio.h>
#include <stdlib.h>
struct node {
  int data;
  struct node* next;
}
struct node* head = NULL;
void insertatbeginning (int value);
void insertatend (int value);
void insertatmiddle (int value, int pos);
void deletefrombeginning ();
void deletefromend();
void deletefrommiddle (int pos);
void displaylist ();
int main(){
  int choice, value, pos;
  printf("\n Menu: \n");
  printf("1. Insert at beginning \n");
  printf("2. Insert at end \n");
  printf("3. Insert at middle \n");
  printf("4. Delete from beginning \n");
  printf("5. Delete from end \n");
  printf("6. Delete from middle \n");
  printf("7. Display list \n");
  printf("8. Exit\n");
  printf("Enter your choice: ");
  scanf("%d", &choice);
  switch(choice) {
  case1: printf("Enter the value to insert: ");
         scanf("%d", &value);
         insertatbeginning(value);
         break;
  case2: printf("Enter the value to insert: ");
         scanf("%d", &value);
         insertatend(value);
         break;
```

```
        case 3: printf("Enter the value to insert: ");
                scanf("%d", &value);
                printf("Enter the position to insert: ");
                scanf("%d", &pos);
                insertatmiddle(value, pos);
                break;
        case 4: deletefrombeginning();
                break;
        case 5: deletefromend();
                break;
        case 6: deletefrom
                printf("Enter the position to delete: ");
                scanf("%d", &pos);
                deletefrommiddle(pos);
                break;
        case 7: displaylist();
                break;
        case 8: printf("Exit");
                exit(0);
        default: printf("Invalid choice! \n");
    }
    displaylist();
    return 0;
}
void insertatbeginning(int value){
    struct node* newnode = (struct node*) malloc(sizeof(struct node));
    newnode->data = value;
    newnode->next = head;
    head = newnode;
    printf("Insertion at beginning successful\n");
}
void insertatend(int value) {
    struct node* newnode = (struct node*) malloc(sizeof(structnode));
    newnode->data = value;
    newnode->next = NULL;
    if (head == NULL) {
        head = newnode;
    }
    else {
        struct node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
```

```
}
printf("Insertion at end successful! \n");
}
void insertatmiddle(int value, int pos){
if(pos<=0){
printf("Invalid position");
return;
}
struct node* newnode = (struct node*) malloc(sizeof(struct node));
newnode->data = value;
if(pos==1){
newnode->next=head;
head=newnode;
}
else{
struct node* temp=head;
for(int i=1; i<pos-1 && temp!=NULL; i++){
displaylist() temp=temp->next;
}
if(temp==NULL){
printf("Invalid pos");
free(newnode);
return;
}
newnode->next=temp->next;
temp->next=newnode;
}
printf("Insertion at middle successful! \n");
}
void deletefrombeginning(){
if(head==NULL){
printf("List is empty! \n");
return;
}
struct node* temp=head;
head=head->next;
free(temp);
printf("Deletion from beginning successful! \n");
}
void deletefromend(){
```

Empty
NLD
22/11/24

```
void deletefrom middle (int pos
    if (pos<=0 || head == NULL) {
        printf ("Empty \n");
        return;
    }
    if (pos == 1) {
        struct node* temp = head;
        head = head -> next;
        free(temp);
        return;
    }
    struct node * temp = head;
    for (int i=1; i<pos-1; i++)
        temp = temp -> next;
    if (temp == NULL || temp -> next == NULL) {
        printf ("Invalid");
        return;
    }
    struct node* temp1 = temp -> next;
    temp -> next = temp -> next -> next;
}
void displaylist() {
    if (head == NULL) {
        printf ("Empty" \n);
        return;
    }
    struct node * temp = head;
    while(temp != NULL) {
        printf ("%d ", temp -> data);
        temp = temp -> next;
    }
    printf (" NULL \n");
}
```

Output:

Single Linked List Operation Menu.

1. Insert at beginning
2. Insert at end
3. Insert at position
4. Delete from beginning

5. Delete from end
6. Delete from position.
7. Exit

Enter your choice: 1
Enter value: 1
1 NULL
Enter your choice: 1
Enter value: 2
2 1 NULL
Enter your choice: 2
Enter value: 3
2 1 3 NULL
Enter your choice: 3
Enter value: 4
Enter position: 2
2 4 1 3 NULL
Enter your choice: 4
4 1 3 NULL
Enter your choice: 5
4 1 NULL
Enter your choice: 6
Enter position: 2
4 NULL
Enter your choice: 7
Exit!

22/1/24

Programiz
C Online Compiler

main.c

Save Run

```c
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    int choice, value, position;
    printf("\nMenu:\n");
    printf("1. Insert at the beginning\n");
    printf("2. Insert at the end\n");
    printf("3. Insert at middle\n");
    printf("4. Delete from beginning\n");
    printf("5. Delete from end\n");
    printf("6. Delete from middle\n");
    printf("7. Exit\n");

    while (1) {
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter the value to insert: ");
                scanf("%d", &value);
                insertAtBeginning(value);
                break;

            case 2:
                printf("Enter the value to insert: ");
                scanf("%d", &value);
                insertAtEnd(value);
```

Output

```
Menu:
1. Insert at the beginning
2. Insert at the end
3. Insert at middle
4. Delete from beginning
5. Delete from end
6. Delete from middle
7. Exit
Enter your choice: 1
Enter the value to insert: 2
Node inserted at the beginning.
2 -> NULL
Enter your choice: 1
Enter the value to insert: 3
Node inserted at the beginning.
3 -> 2 -> NULL
Enter your choice: 2
Enter the value to insert: 4
Node inserted at the end.
3 -> 2 -> 4 -> NULL
Enter your choice: 3
Enter the value to insert: 5
Enter the position to insert: 2
Node inserted at position 2.
3 -> 5 -> 2 -> 4 -> NULL
Enter your choice: 4
Node deleted from the beginning.
5 -> 2 -> 4 -> NULL
Enter your choice: 5
Node deleted from the end.
5 -> 2 -> NULL
Enter your choice: 6
Enter the position to delete: 2
```