Singly Linked List

1. **C Program to delete node from the beginning of Singly Linked List.**

```
#include <stdio.h>
#include <stdlib.h>
struct node {
  int data;
  struct node *next;
}*head;
void createList(int n);
void deleteFirstNode();
void displayList();
void main()
{
  int n;
  printf("Enter the total number of nodes: ");
  scanf("%d", &n);
  createList(n);

  printf("\nData in the list \n");
  displayList();

  printf("\nto delete first node: ");

    deleteFirstNode();

  printf("\nData in the list \n");
  displayList();

  getch();
}
```

```
void createList(int n)
{
    struct node *newNode, *temp;
    int data, i;

    head = (struct node *)malloc(sizeof(struct node));

    if(head == NULL)
    {
        printf("Unable to allocate memory.");
    }
    else
    {

        printf("Enter the data of node 1: ");
        scanf("%d", &data);

        head->data = data;
        head->next = NULL;
        temp = head;

        for(i=2; i<=n; i++)
        {
            newNode = (struct node *)malloc(sizeof(struct node));
            if(newNode == NULL)
            {
                printf("Unable to allocate memory.");
                break;
            }
            else
            {
                printf("Enter the data of node %d: ", i);
                scanf("%d", &data);

                newNode->data = data;
                newNode->next = NULL;

                temp->next = newNode;
                temp = temp->next;
            }
        }

        printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
    }
}

void deleteFirstNode()
{
    struct node *toDelete;

    if(head == NULL)
    {
        printf("List is already empty.");
    }
    else
    {
        toDelete = head;
        head = head->next;

        printf("\nData deleted = %d\n", toDelete->data);
        free(toDelete);

        printf("SUCCESSFULLY DELETED FIRST NODE FROM LIST\n");
    }
}
void displayList()
{
    struct node *temp;

    if(head == NULL)
   {
        printf("List is empty.");
    }
    else
    {
        temp = head;
        while(temp != NULL)
        {
            printf("Data = %d\n", temp->data);
            temp = temp->next;
        }
    }
}
```

2 C Program to delete node from the End of Singly Linked List.

```
#include <stdio.h>

#include <stdlib.h>
```

```
struct node {
    int data;
    struct node *next;
}*head;

void createList(int n);
void deleteLastNode();
void displayList();


void main()
{
    int n;
    printf("Enter the total number of nodes: ");
    scanf("%d", &n);
    createList(n);

    printf("\nData in the list \n");
    displayList();

    printf("\nto delete last node: ");
```

```
        deleteLastNode();



    printf("\nData in the list \n");

    displayList();



getch();

}

void createList(int n)

{

    struct node *newNode, *temp;

    int data, i;

    head = (struct node *)malloc(sizeof(struct node));

    if(head == NULL)

    {

        printf("Unable to allocate memory.");

    }

    else

    {



        printf("Enter the data of node 1: ");

        scanf("%d", &data);
```

```
head->data = data;

head->next = NULL;

temp = head;



for(i=2; i<=n; i++)

{

    newNode = (struct node *)malloc(sizeof(struct node));



    if(newNode == NULL)

    {

        printf("Unable to allocate memory.");

        break;

    }

    else

    {

        printf("Enter the data of node %d: ", i);

        scanf("%d", &data);



        newNode->data = data;

        newNode->next = NULL;
```

```
                temp->next = newNode;

                temp = temp->next;

            }

        }



        printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");

    }

}

void deleteLastNode()

{

    struct node *toDelete, *secondLastNode;



    if(head == NULL)

    {

        printf("List is already empty.");

    }

    else

    {

        toDelete = head;

        secondLastNode = head;
```

```
                while(toDelete->next != NULL)
            {
                secondLastNode = toDelete;
                toDelete = toDelete->next;
            }

            if(toDelete == head)
            {
                head = NULL;
            }
            else
            {
                secondLastNode->next = NULL;
            }
        free(toDelete);
        printf("SUCCESSFULLY DELETED LAST NODE OF LIST\n");
          }
        }
        void displayList()
        {
          struct node *temp;
```

```
    if(head == NULL)
    {
        printf("List is empty.");
    }
    else
    {
        temp = head;
        while(temp != NULL)
        {
            printf("Data = %d\n", temp->data);
            temp = temp->next;
        }
    }
}
```

3 **C Program to delete node from the Middle of Singly Linked List.**

```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node *next;
} *head;
```

```
void createList(int n);

void deleteMiddleNode(int position);

void displayList();

void main()

{

    int n, position;

 printf("Enter the total number of nodes: ");

    scanf("%d", &n);

    createList(n);

 printf("\nData in the list \n");

    displayList();

    printf("\nEnter the node position you want to delete: ");

    scanf("%d", &position);

    deleteMiddleNode(position);


    printf("\nData in the list \n");

    displayList();


    getch();

}
```

```
void createList(int n)

{

    struct node *newNode, *temp;

    int data, i;



    head = (struct node *)malloc(sizeof(struct node));

    if(head == NULL)

    {

        printf("Unable to allocate memory.");

    }

    else

    {

        printf("Enter the data of node 1: ");

        scanf("%d", &data);



        head->data = data;

        head->next = NULL;

        temp = head;



        for(i=2; i<=n; i++)
    {
        newNode = (struct node *)malloc(sizeof(struct node));


        if(newNode == NULL)
        {
            printf("Unable to allocate memory.");
            break;
        }
        else
        {
            printf("Enter the data of node %d: ", i);
            scanf("%d", &data);


            newNode->data = data;
            newNode->next = NULL;
            temp->next = newNode;
            temp = temp->next;
        }
    }
printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
 }
```

```
}

void deleteMiddleNode(int position)

{

    int i;

    struct node *toDelete, *prevNode;



    if(head == NULL)

    {

        printf("List is already empty.");

    }

    else

    {

        toDelete = head;

        prevNode = head;



        for(i=2; i<=position; i++)

        {

            prevNode = toDelete;

            toDelete = toDelete->next;



            if(toDelete == NULL)
```

```
                break;
            }

            if(toDelete != NULL)
            {
                if(toDelete == head)
                    head = head->next;

                prevNode->next = toDelete->next;
                toDelete->next = NULL;
                free(toDelete);

                printf("SUCCESSFULLY DELETED NODE FROM MIDDLE OF LIST\n");
            }
            else
            {
                printf("Invalid position unable to delete.");
            }
        }
    }
```

```
void displayList()
{
    struct node *temp;
    if(head == NULL)
    {
        printf("List is empty.");
    }
    else
    {
        temp = head;
        while(temp != NULL)
        {
            printf("Data = %d\n", temp->data);
            temp = temp->next;
        }
    }
}
```