```c
1.//Linked List Having three element
#include <stdio.h>
#include<stdlib.h>

struct Node{
    int data;
    struct Node *next;
};
void display(struct Node *);

int main(){
    struct Node *head;


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

    //Node2
    struct Node *ptr ;
    ptr = (struct Node*)malloc(sizeof(struct Node));
    //Node 3

    struct Node *node3 = (struct Node*)malloc(sizeof(struct Node));

    //Created a single node........

    //head = 0x200;
    head->data=10;
    head->next=ptr;

    //Node 2
    ptr->data = 20;
    ptr->next = node3;

    //Node3

    node3->data = 30;
    node3->next = NULL;


    display(head);

    return 0;
}

void display(struct Node *p){
    //p = 0x200;
    while(p!=NULL)
    {
        printf("%d->",p->data);
```

```
        p=p->next;//Now p=NULL;
    }
}
```

2.

```
//Linked list displayed using recursion......
//Linked List Having three element
#include <stdio.h>
#include<stdlib.h>

struct Node{
    int data;
    struct Node *next;
};
void display(struct Node *);
void RDisplay(struct Node *);
int NCount(struct Node *);

int main(){
    struct Node *head;


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

    //Node2
    struct Node *ptr ;
    ptr = (struct Node*)malloc(sizeof(struct Node));
    //Node 3

    struct Node *node3 = (struct Node*)malloc(sizeof(struct Node));

    //Created a single node........

    //head = 0x200;
    head->data=10;
    head->next=ptr;

    //Node 2
    ptr->data = 20;
    ptr->next = node3;

    //Node3

    node3->data = 30;
    node3->next = NULL;
```

```
    //display(head);

    //Recursive function call
    RDisplay(head);
    int n =NCount(head);
    printf("\nThe number of nodes  = %d",n);

    return 0;
}

void display(struct Node *p){
    //p = 0x200;
    while(p!=NULL)
    {
        printf("%d->",p->data);
        p=p->next;//Now p=NULL;
    }
}
void RDisplay(struct Node *p){
    if(p!=NULL){
        printf("%d->",p->data);
        RDisplay(p->next);
    }
}
int NCount(struct Node *p){

    int count=0;
    while(p){
        count++;
        p=p->next;
    }
    return count;
}
```

3.

```
//Linked List Node Counting using Recursion.................
#include <stdio.h>
#include<stdlib.h>

struct Node{
    int data;
    struct Node *next;
};
void display(struct Node *);
void RDisplay(struct Node *);
int NCount(struct Node *);
```

```c
int RCount(struct Node *);
int Sum(struct Node *);
int SumII(struct Node *);


int main(){
    struct Node *head;


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

    //Node2
    struct Node *ptr ;
    ptr = (struct Node*)malloc(sizeof(struct Node));
    //Node 3

    struct Node *node3 = (struct Node*)malloc(sizeof(struct Node));

    //Created a single node........

    //head = 0x200;
    head->data=10;
    head->next=ptr;

    //Node 2
    ptr->data = 20;
    ptr->next = node3;

    //Node3

    node3->data = 30;
    node3->next = NULL;


    //display(head);

    //Recursive function call
    RDisplay(head);
    //Counting the number of nodes..
    int n =NCount(head);
    printf("\nThe number of nodes  = %d",n);
    int i = RCount(head);
    printf("\nBy Recursion : The number of nodes   = %d",i);
    int s = Sum(head);
    printf("\nThe sum of data in the LL using recursion = %d",s);
    int x =SumII(head);
    printf("\nThe sum of data in the LL using fn call = %d",x);
    return 0;
}
```

```c
void display(struct Node *p){
    //p = 0x200;
    while(p!=NULL)
    {
        printf("%d->",p->data);
        p=p->next;//Now p=NULL;
    }
}
void RDisplay(struct Node *p){
    if(p!=NULL){
        printf("%d->",p->data);
        RDisplay(p->next);
    }
}
int NCount(struct Node *p){

    int count=0;
    while(p){
        count++;
        p=p->next;
    }
    return count;
}
//Recursive function for counting the number of nodes
int RCount(struct Node *p){
    if(p==NULL){
        return 0;
    }else{
        return RCount(p->next)+1;
    }
}
//Sum of data in LL
int Sum(struct Node *p)
{
    if(p==NULL){
        return 0;
    }else{


        return Sum(p->next)+p->data;
    }

}

//Sum of data without recursion

int SumII(struct Node *p)
```

```
{
    int sum= 0;
    while(p){
        sum += p->data;
        p=p->next;
    }
    return sum;

}
```

4.

```
//Max element from LL usinng recursion.............
#include<stdio.h>
#include<stdlib.h>
struct Node
{
    int data;
    struct Node*next;
};
void display(struct Node*);
int max(struct Node*);
int Rmax(struct Node*);
int main()
{
    struct Node *head;
    struct Node *first=(struct Node*)malloc(sizeof(struct Node));
    struct Node *second=(struct Node*)malloc(sizeof(struct Node));
    struct Node *third=(struct Node*)malloc(sizeof(struct Node));
    head=first;
    first->data=10;
    first->next=second;
    second->data=120;
    second->next=third;
    third->data=30;
    third->next=NULL;
    display(first);
    printf("\n");
    int maximum=max(first);
    printf("maximum value is %d \n",maximum);
    int Rmaximum=Rmax(first);
    printf("maximum value is %d \n",Rmaximum);
    return 0;
}
void display(struct Node *p)
{
    while(p!=NULL)
    {
```

```
        printf("%d-->",p->data);
        p=p->next;
    }
}
int max(struct Node *p)
{
    int m=-32768;
    while(p!=NULL)
    {
        if((p->data)>m)
        {
            m=p->data;
        }
        p=p->next;
    }
    return m;
}
int Rmax(struct Node *p)
{
    int x=0;
    if(p==0)
    {
        return 0;
    }
    else
    {
        x=max(p->next);
        if(x>p->data)
        {
            return x;
        }
        else
        {
            return p->data;
        }
    }
}
```

5.

```
//searching for an element in the linkedlist
#include<stdio.h>
#include<stdlib.h>
struct Node
{
int data;
struct Node *next;
};
```

```c
void display(struct Node*);
struct Node* search(struct Node*,int);
int main()
{
struct Node *head;
head=(struct Node*)malloc(sizeof(struct Node));
//head->data=10;
struct Node*first=(struct Node*)malloc(sizeof(struct Node));
head->next=first;
first->data=10;
struct Node*second=(struct Node*)malloc(sizeof(struct Node));;
second->data=20;
first->next=second;
struct Node*third=(struct Node*)malloc(sizeof(struct Node));;
third->data=50;
second->next=third;
third->next=NULL;
display(first);
printf("\n");
//int key=20;
struct Node*temp;
temp=search(first,20);
printf("found %d",temp->data);
return 0;
}
void display(struct Node*p)
{
while(p!=NULL)
{
printf("%d-->",p->data);
p=p->next;
}
}
struct Node* search(struct Node*p,int key)
{
while(p!=NULL)
{
if(key==p->data)
{
return p;
}
p=p->next;
}
return NULL;
}
```

6.

```c
. #include <stdio.h>
#include <stdlib.h>
#include <limits.h>
typedef struct Node {
int data;
struct Node *next;
}Node;
Node *head = NULL;
void display1(Node *);
void display2(Node *);
int nCount(Node *);
int rCount(Node *);
int nSum(Node *);
int rSum(Node *);
int nMax(Node *);
int rMax(Node *);
Node* nSearch(Node *, int);
void insert(Node *, int, int);
int main() {
head = (Node *)malloc(sizeof(Node));
head->data = 20;
head->next = (Node *)malloc(sizeof(Node));
head->next->data = 10;
head->next->next = (Node *)malloc(sizeof(Node));
head->next->next->data = 30;
head->next->next->next = NULL;
/*printf("%d\n", head->data);
printf("%d\n", head->next->data);
printf("%d\n", head->next->next->data);*/
printf("Original list: ");
display1(head);
printf("\n");
printf("Reversed list: ");
display2(head);
printf("\nNumber of nodes (iteration): %d\n", nCount(head));
printf("Number of nodes (recursion): %d\n", rCount(head));
printf("Sum of elements (iteration): %d\n", nSum(head));
printf("Sum of elements (recursion): %d\n", rSum(head));
printf("Max of elements (iteration): %d\n", nMax(head));
printf("Max of elements (recursion): %d\n", rMax(head));
Node *key = nSearch(head, 20);
printf("Element found: %d\n", key->data);
insert(head, 0, 40);
display1(head);
printf("\n");
insert(head, 3, 50);
display1(head);
return 0;
```

```
}
//function to display using recursion
void display1(Node *p) {
if (p != NULL) {
printf("%d -> ", p->data);
display1(p->next);
}
}
//function to display using recursion but in reverse order
void display2(Node *p) {
if (p != 0) {
display2(p->next);
printf("%d <- ", p->data);
}
}
//function to count the number of nodes in the linked list
int nCount(Node *p) {
int c = 0;
while (p) {
c++;
p = p->next;
}
return c;
}
//function to count using recursion
int rCount(Node *p) {
if (p == 0) {
return 0;
} else {
return 1 + rCount(p->next);
}
}
//function to find sum using iteration
int nSum(Node *p) {
int sum = 0;
while (p) {
sum += p->data;
p = p->next;
}
return sum;
}
//function to find sum using recursion
int rSum(Node *p) {
int sum = 0;
if (!p) {
return 0;
} else {
sum += p->data;
```

```
return sum + rSum(p->next);
}
}
//function to find maximum using iteration
int nMax(Node *p) {
int max = INT_MIN;
while(p != NULL) {
if((p->data) > max) {
max = p->data;
}
p = p->next;
}
return max;
}
//function to find max using recursion
int rMax(Node *p) {
int max = INT_MIN;
if (p == 0) {
return INT_MIN;
}
else {
max = rMax(p->next);
if(max > p->data)
return max;
else
return p->data;
}
}
//function to find the element
Node* nSearch(Node *p, int key) {
while(p != NULL) {
if(key == p->data)
return p;
p = p -> next;
}
return NULL;
}
//to nsert at a position
void insert(Node *p, int index, int x) {
Node *t;
int i;
if(index < 0 || index > nCount(p)) {
printf("\nInvalid position!");
}
t = (Node*)malloc(sizeof(Node));
t->data = x;
if(index == 0) {
t->next = head;
```

```
head = t;
} else {
for(i = 0; i < index-1; i++) {
p = p->next;
}
t->next = p->next;
p->next = t;
}
}
```