LINKED LIST

```c
#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));
    head->data=10;
    head->next=NULL;
    display(head);
    return 0;
}
void display(struct Node *p){
    while(p!=NULL){
        printf("%d->",p->data);
        p=p->next;
    }
}
```

Output

```
10->
```

```c

#include<stdio.h>
#include<stdlib.h>
struct Node
{
    int data;
    struct Node *next;
}*head=NULL;
void RDisplay(struct Node*);
void display(struct Node*);
int NCount(struct Node*);
int RCount(struct Node*);
int Nsum(struct Node*);
int RSum(struct Node*);
int MaxElement(struct Node*);
int RMaxElement(struct Node*);
struct Node* Lsearch(struct Node *,int);
void insert(struct Node *,int,int);
void create(int *,int n);
int main()
{
    struct Node *temp;
    int A[]={10,20,30,40,50};
    create(A,5);
    // struct Node *temp;
    // head=(struct Node*)malloc(sizeof(struct Node));
    // n1=(struct Node*)malloc(sizeof(struct Node));
    // n2=(struct Node*)malloc(sizeof(struct Node));
    // head->data=10;
    // head->next=n1;
    // n1->data=50;
    // n1->next=n2;
    // n2->data=40;
    // n2->next=NULL;
    display(head);
    printf("\n");
    RDisplay(head);
    printf("\n");
    // int count= NCount(head);
    int count= RCount(head);
    int sum=Nsum(head);
    int r_sum=RSum(head);
    int max=MaxElement(head);
    int r_max=RMaxElement(head);
    printf("Total number of node=%d\n",count);
    printf("Total sum=%d\n",sum);
    printf("Total sum=%d\n",r_sum);
    printf("Maximum element =%d\n",max);
    printf("Maximum element =%d\n",r_max);
    temp=Lsearch(head,50);
    printf("%d\n",temp->data);
    insert(head,0,5);
    RDisplay(head);
    printf("\n");
    insert(head,2,15);
    RDisplay(head);
    return 0;
}
void display(struct Node *p)
{
    while(p!=NULL)
    {
        printf("%d->",p->data);
        p=p->next;
    }
}
void RDisplay(struct Node *p)
{
    if(p!=NULL)
    {
        RDisplay(p->next);
        printf("%d->",p->data);

    }
}
int NCount(struct Node *p)
{
    int c=0;
    while(p)
    {
        c++;
        p=p->next;
    }
    return c;
}
int RCount(struct Node*p)
{
    if(p==NULL)
    {
        return 0;
```

```c
    }
    else
    {
        return RCount(p->next)+1;
    }
}
int Nsum(struct Node *p)
{
    int s=0;
    while(p!=NULL)
    {
        s=s+p->data;
        p=p->next;
    }
    return s;
}
int RSum(struct Node*p)
{
    if(p==NULL)
    {
        return 0;
    }
    else
    {
        return RSum(p->next)+p->data;
    }
}
int MaxElement(struct Node *p)
{
    int m=-32768;
    while(p!=NULL)
    {
        if(p->data > m)
        {
            m=p->data;
        }
        p=p->next;
    }
    return m;
}
int RMaxElement(struct Node *p)
{
    int x=0;
    if(p==0)
    {
        return -32768;
    }
    else
    {
        x=RMaxElement(p->next);
        if(x> p->data)
        {
            return x;
        }
        else
        {
            return p->data;
        }
    }
}
struct Node* Lsearch(struct Node *p,int key)
```

```c
{
    while(p!=NULL)
    {
        if(key==p->data)
        {
            return p;
        }
        p=p->next;
    }
    return NULL;
}
void insert(struct Node *p,int index,int x)
{
    struct Node*t;
    int i;
    if(index<0 || index>NCount(p))
    {
        printf("Invalidposition \n");
    }
    t=(struct Node *)malloc(sizeof(struct 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;
    }
}
void create(int A[],int n)
{
    struct Node *t,*last;
    head=(struct Node*)malloc(sizeof(struct Node));
    head->data=A[0];
    head->next=NULL;
    last=head;
    for(int i=1;i<n;i++)
    {
        t=(struct Node*)malloc(sizeof(struct Node));
        t->data=A[i];
        t->next=NULL;
        last->next=t;
        last=t;
    }
}
```

Output

```
10->20->30->40->50->
50->40->30->20->10->
Total number of node=5
Total sum=150
Total sum=150
Maximum element =50
Maximum element =50
50
50->40->30->20->10->5->
50->40->30->20->15->10->5->
```