LINKED LIST

```
#include <stdio.h>

#include<stdlib.h>

struct Node {

   int data;

   struct Node* next;


};

void display(struct Node*);

int main() {

   struct Node *head,*n1,*n2;

   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=20;

   n1->next=n2;

   n2->data=40;

   n2->next=NULL;

   display(head);

   return 0;


}

void display(struct Node *p) {

   while( p != NULL) {

      printf("%d->",p->data);

      p=p->next;


   }

   }
```

Sum of elements

```
#include <stdio.h>

#include<stdlib.h>

struct Node {

    int data;

    struct Node* next;


};

void display(struct Node*);

int sum(struct Node*);

int Rsum(struct Node*);

int main() {

    struct Node *head,*n1,*n2;

    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=20;

    n1->next=n2;

    n2->data=40;

    n2->next=NULL;

    display(head);

    int s = Rsum(head);

    printf("Sum of elements = %d",sum);

    return 0;


}

void display(struct Node *p) {

    while( p != NULL) {

        printf("%d->",p->data);
        p=p->next;


    }


}
int sum(struct Node*p){
    int sum=0;
    while(p!=NULL){
        sum+=p->data;
        p=p->next;
    }
    return sum;
}
int Rsum(struct Node*p){
    if(p==NULL){
        return 0;
    }else{
        return Rsum(p->next)+p->data;
    }
}



#include <stdio.h>
#include<stdlib.h>
struct Node {
    int data;
    struct Node *next;

}*head = NULL;
void display(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);

int main() {

    struct Node *temp;

    int A[] = {10,20,30,40,50};

    create(A,5);

    display(head);

    int max=RMaxElement(head);

    printf("\nMaximum element is %d",max);

    temp=Lsearch(head,20);

    printf("\n%d",temp->data);

    insert(head,0,5);

    printf("\n");

    display(head);

    return 0;


}

void display(struct Node *p) {

    while( p != NULL) {

        printf("%d->",p->data);

        p=p->next;


    }


}

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;
        }
    }
}
struct Node * Lsearch(struct Node* p,int key){
    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){
        printf("Invalid position\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;
    }
}
```