PHASE 1 DAY 21

LINKED LIST

```
#include <stdio.h>
#include<stdlib.h>

struct Node
{
    int data;
    struct Node *next;

};
void display(struct Node*);
void Recursivedisplay(struct Node*);
int Rcount(struct Node*p);
int count(struct Node *);
int sum(struct Node*);
int Rsum(struct Node *);
int max(struct Node *);
int Rmax(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=500;
  head->next=n1;

  n1->data=2;
  n1->next=n2;

  n2->data=549;
  n2->next=NULL;

 printf("1.display function:\n") ;
  display(head);
    printf("NULL");
printf("\n2.Recursive display function:\n") ;
 Recursivedisplay(head);
   printf("NULL");

 printf("\n3.count function: \n") ;
 int result=count(head);
 printf("count=%d",result);

printf("\n4.Recursive count function: \n") ;
 int result1=Rcount(head);
 printf("count=%d",result1);

printf("\n5.Sum function: \n") ;
 int result2=sum(head);
 printf("sum=%d",result2);

printf("\n6.Recursive sum function: \n") ;
 int result3=Rsum(head);
 printf("sum=%d",result3);

 printf("\n7. max function: \n") ;
 int result4=max(head);
 printf("Max=%d",result4);

 printf("\n8.Recursion max function: \n") ;
 int result5=Rmax(head);
 printf("Max=%d",result5);

 return 0;
}

void display(struct Node *p)
{

    while(p!=NULL)
    {
        printf("%d->",p->data);
        p=p->next;
    }

}

void Recursivedisplay(struct Node *p)
{
     if(p!=NULL)
     {
         printf("%d->",p->data);
         Recursivedisplay(p->next);

     }
}

int count(struct Node*p)
{
    int c=0;

    while(p!=NULL)
    {
        c++;
        p=p->next;
    }
    return c;
}

 int Rcount(struct Node*p)
{

    if(p==NULL)
    {
    return 0;
    }
  else
    {
       return  Rcount(p->next)+1;
    }
}

 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;
    }
}
 int max(struct Node *p)
 {
     int max= p->data;
     while(p!=NULL)
     {
         if(max < p->data)
         {
             max=p->data;
         }
         p=p->next;

     }

     return max;
 }
 int Rmax(struct Node *p)
 {
     int x=0;

     if (p==0)
     return -327;

     else
     {
     x=Rmax(p->next);

     if(x>p->data)
     return x;
     }

 }
```

//search function

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

Find element

```
//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;
    }
}
```