SINGLY LINKEDLIST

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

typedef struct node{
   int data;
   struct node *next;
}node;

void display(node *p);
void rDisplay(node *p);
void reverse_list(node *p);
void Count(node *p);
int rCount(node *p);
void Sum(node *p);
int rSum(node *p);
int maximum(node *p);
int minimum(node *p);
int rMaximum(node *p);
int rMinimum(node *p);
node* search_key(node *p,int);

int main(){
   node *head,*n1,*n2,*n3,*n4;

   head=(node*)malloc(sizeof(node));
   n1=(node*)malloc(sizeof(node));
   n2=(node*)malloc(sizeof(node));
   n3=(node*)malloc(sizeof(node));
   n4=(node*)malloc(sizeof(node));
```

```
head->data=1000;
head->next=n1;

n1->data=-200;
n1->next=n2;

n2->data=3000;
n2->next=n3;

n3->data=90;
n3->next=n4;

n4->data=500;
n4->next=NULL;

printf("\n\nDisplay linked list using normal function display()\n");
display(head);

printf("\n\nDisplay linked list using recursive function rDisplay()\n");
rDisplay(head);

printf("\n\nDisplay reversed linked list using recursive function reverse_list()\n");
reverse_list(head);

printf("\n\nDisplay number of nodes in linked list using  function Count()\n");
Count(head);

printf("\n\nDisplay number of nodes in linked list using  recursive function rCount()\n");
int c=rCount(head);
printf("count = %d",c);
```

```
printf("\n\nDisplay sum of nodes in linked list using   function Sum()\n");
Sum(head);

printf("\n\nDisplay sum of nodes in linked list using  recursive function rSum()\n");
int s=rSum(head);
printf("sum = %d",s);

printf("\n\nDisplay maximum node value in linked list using  function rSum()\n");
int m=maximum(head);
printf("maximum = %d",m);

printf("\n\nDisplay maximum node value in linked list using  function rSum()\n");
int max=rMaximum(head);
printf("maximum value = %d",max);

printf("\n\nDisplay minimum node value in linked list using  function rSum()\n");
int n=minimum(head);
printf("minimum = %d",n);

printf("\n\nDisplay minimum node value in linked list using  function rSum()\n");
int min=rMinimum(head);
printf("minimum value = %d",min);


printf("\n\nSearching for a number:\n");
node *res=search_key(head,-200);
printf("%d found at %p location",*res,res);


}
```

```
void display(node *p){
   while(p!=NULL){

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

   }printf("NULL");
}

void rDisplay(node *p){
   if(p!=0){
      printf("%d->",p->data);
      rDisplay(p->next);
   }
}

void reverse_list(node *p){
   if(p!=0){
      reverse_list(p->next);
      printf("%d->",p->data);
   }
}

void Count(node *p){
   int count=0;
   while(p!=0){
      count++;
      p=p->next;
   }
   printf("Count = %d",count);
}
```

```
int rCount(node *p){
   if(p==0){
      return 0;
   }else{
      return rCount(p->next)+1;
   }
}

void Sum(node *p){
   int sum=0;
   while(p!=NULL){
      sum=sum+p->data;
      p=p->next;
   }
   printf("sum = %d ",sum);
}

int rSum(node *p){
   if(p==0){
      return 0;
   }else{
      return p->data+rSum(p->next);
   }
}

int maximum(node *p){
   int max=p->data;
   while(p!=NULL){

      if(p->data>max){
```

```
            max=p->data;

        } p=p->next;

    }return max;
}

int rMaximum(node *p){
    if(p==0){
        return 0;
    }else{
        int x=rMaximum(p->next);
        if(x>p->data){
            return x;
        }
    }

}

int minimum(node *p){
    int min=p->data;
    while(p!=NULL){
        if(p->data<min){
            min=p->data;
        }p=p->next;
    }return min;
}

int rMinimum(node *p){
    if(p==0){
        return 0;
```

```
    }else{
        int min=rMinimum(p->next);
        if(min<p->data){
            return min;
        }

    }
}

node* search_key(node *p,int key){

    while(p!=NULL){
        if(key==p->data){
            return p;

        }p=p->next;
    }return NULL;
}
```