Day 21

1. //linkedlist

```
#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;
   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);//printing the node creating
   return 0;
}
void display(struct Node*p)
```

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


   }


}
```

```
PS D:\learning c> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day21-2.exe'
10-->20-->50-->
PS D:\learning c\output>
```

2.

```
//using recursive function
//linkedlist
#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;
   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);//printing the node creating
    return 0;
}
void display(struct Node*p)
{
    if(p!=NULL)
    {
        printf("%d-->",p->data);
        display(p->next);

    }

}
```

```
PS D:\learning c> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day21-2.exe'
10-->20-->50-->
PS D:\learning c\output>
```

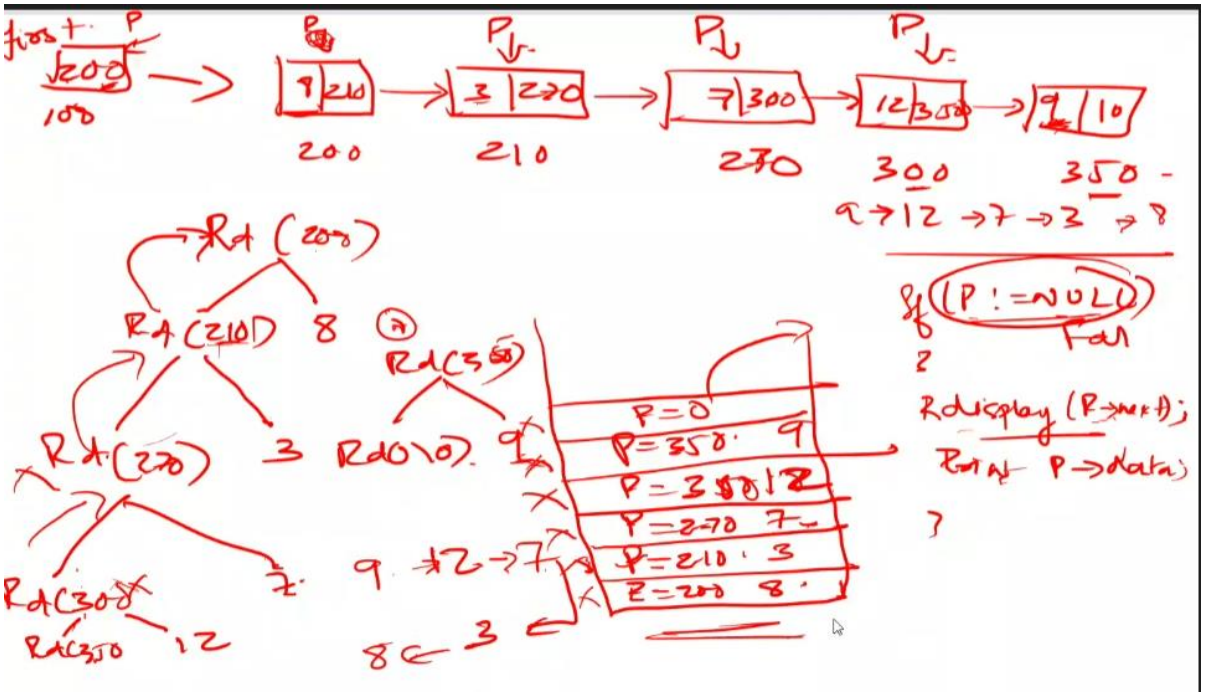
3. //using recursive function

```
//linkedlist
#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;
   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);//printing the node creating
   return 0;
}
void display(struct Node*p)
{
```

```
    if(p!=NULL)
    {
        display(p->next);//recursive call first then printing
        printf("%d-->",p->data);
        //this is the output now:50-->20-->10-->



    }



}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day21-3.exe'
50-->20-->10-->
PS D:\learning c\output>
```

Why?



4. //counting no.of nodes in a linkedlist

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

   count(first);
   return 0;
}
void count(struct Node*p)
{
```

```
    int count=0;
    while(p!=NULL)
    {
        count++;
        p=p->next;



    }
    printf("count is %d",count);



}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day21-4.exe'
count is 3
```

5. //counting no.of nodes in a linkedlist

```
//using recursion
#include<stdio.h>
#include<stdlib.h>
struct Node
{
    int data;
    struct Node *next;
};
int count(struct Node*);
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;
    //int c=0;
   int c= count(first);
   printf("c is %d",c);
    return 0;
}
int count(struct Node*p)
{

    if(p==0)
    {
        return 0;
    }
    else if(p!=NULL)
    {

        return count(p->next)+1;



    }
```

}

```
PS D:\learning c> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day21-5.exe'
c is 3
PS D:\learning c\output>
```

6. //sum of all the elememts in a linkedlist with and without recursion'

```
#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;//(struct Node*)malloc(sizeof(struct Node));
   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=20;
```

```
    second->next=third;
    third->data=30;
    third->next=NULL;
    display(first);
    int sumfn=sum(first);
    printf("\n");
    printf("sum is %d \n",sumfn);
    int Rsumfn=Rsum(first);
    printf("using recursion\n");
    printf("sum is %d \n",Rsumfn);
    return 0;
}
void display(struct Node *p)
{
    while(p!=NULL)
    {
        printf("%d-->",p->data);
        p=p->next;
    }
}
int sum(struct Node *s)
{
    int sum=0;
    while(s!=NULL)
    {
       sum+=s->data;
       s=s->next;
    }
    return sum;
}
int Rsum(struct Node *s)
```

```
{

 if(s!=NULL)
  {

    return s->data+Rsum(s->next);
  }

}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day21-6.exe'
10-->20-->30-->
sum is 60
using recursion
sum is 60
PS D:\learning c\output>
```

7. //finding out max element in a linkedlist

```
#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;
       }
    }
}
```

```
PS D:\learning c\output> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day21-7.exe'
10-->120-->30-->
maximum value is 120
maximum value is 120
PS D:\learning c\output>
```

8. //searching for an element in the linkedlist

```
#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;
```

}

```
PS D:\learning c> cd 'd:\learning c\output'
PS D:\learning c\output> & .\'day21-8.exe'
10-->20-->50-->
found  20
PS D:\learning c\output>
```

9. #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;
    }
}
```

```
40 -> 20 -> 10 -> 30 ->
40 -> 20 -> 10 -> 50 -> 30 ->
```