Data Structures

Reversing of a linked list

printing the data reversely:

- Base condition is if head equal to null return nothing.
- Traverse through the end of a linked list recursively.
- print the data of the node after recursive step so that we traverse through until the last node of linked list and then during return we print the data.

```
void PrintElements(lin_list *head){
    //base condition
    if(head==NULL){
        return;
    }
    //recursive step
    PrintElements(head->next);
    printf("%d ",head->data);
}
```

Function for reversing a linked list:

- create a new linked list(tail) and point it to NULL.
- Traverse through each node of linked list.
- Point the next pointer of head to tail.
- push each node to the newly created linked list(tail).

```
/* Reverse the elements in the linked list */
lin_list *ReverseList(lin_list *head) {
    lin_list *next, *tail = NULL;
    while (head) {
        //storing the linked list without first node into next.
        next = head->next;
        head->next = tail;
        tail = head;
        //restoring the linked list into head.
        head = next;
    }
    return tail;
}
```

Whole program:

```
#include<stdio.h>
#include<stdlib.h>
//creating a node.
typedef struct lin_list{
    int data;
    struct lin_list *next;
}lin_list;

//inserting nodes
lin_list *insertnode(lin_list *head,int data) {
    lin_list *newnode=(lin_list*)malloc(sizeof(lin_list));
    newnode->data=data;
    newnode->next=head;
    head=newnode;
    return head;
}
//printing the linked list.
void PrintElements(lin_list *head){
    //base condition
```

```
    if(head==NULL){
        return;
    }
    printf("%d ",head->data);
    PrintElements(head->next);
}

void RecursiveReverse(lin_list *head){
    //base condition
    if(head==NULL){
        return;
    }
    //recursive step
    RecursiveReverse(head->next);
    printf("%d ",head->data);
}

/* Reverse the nodes in the linked list */
lin_list *ReverseList(lin_list *head) {
    lin_list *next, *tail = NULL;
    while (head) {
        //storing the linked list without first node into next.
        next = head->next;
        head->next = tail;
```

```
        tail = head;
        //restoring the linked list into head.
        head = next;
    }
    return tail;
}
int main(){
    lin_list *head=NULL;
    head=insertnode(head,4);
    head=insertnode(head,3);
    head=insertnode(head,2);
    head=insertnode(head,1);
    PrintElements(head);printf("\n");
    //reversing of linked list
    head=ReverseList(head);
    PrintElements(head);printf("\n");
    //printing elements reversely.
    RecursiveReverse(head);
    return 0;
}
```

```
1 2 3 4
4 3 2 1
1 2 3 4
```