```c
#include<stdio.h>
#include<stdlib.h>
typedef struct node
{
    int data;
    struct node *next;
}node;
node *head;
void create();
void display();
void Insert_At_Pos(int data,int pos);
void Delete_All_Nodes();
void reverse();

int main()
{
    int choice;
    while(1)
    {
        printf("\nEnter choice:\n1.Create SLL\n2.Display\n3.Insert at any position\n4.Delete all nodes\n5.Reverse the linked list\n6.Exit\n");
        scanf("%d",&choice);
        int pos,data;
        switch(choice)
        {
            case 1: create();
                    break;
            case 2: display();
                    break;
            case 3: printf("\nEnter Position where you want to enter: ");
                    scanf("%d",&pos);
                    printf("\nEnter Data: ");
                    scanf("%d",&data);
                    Insert_At_Pos(data,pos);
                    break;
            case 4: Delete_All_Nodes();
                    break;
            case 5: reverse();
                    break;
            case 6: return 0;
            default: printf("\nInvalid Input\n");
                     return 0;

        }

    }
}

void create()
{
    int num,i;
    node *new_node,*ptr;
    printf("\nEnter Number of data you want to enter: ");
    scanf("%d",&num);
    head=ptr=NULL;
    while(num)
    {
        new_node=(node *)malloc(sizeof(node));
        printf("\nEnter data: ");
        scanf("%d",&new_node->data);
        new_node->next=NULL;
        if(head==NULL)
            head=ptr=new_node;
        else
        {
            ptr->next=new_node;
            ptr=ptr->next;
        }
        num--;
    }
    printf("\nLinked List Created\n");
    printf("\n_____________________________________________________________________________\n");
}

void display()
{
    node *ptr;
    ptr=head;
    if(head==NULL)
        {
            printf("\nThe list is empty\n");
            return ;
        }
    printf("\nDisplaying Elements of Linked List: ");
    while(ptr)
    {
        printf("%d ",ptr->data);
        ptr=ptr->next;
    }
    printf("\n_____________________________________________________________________________\n");
}

void Delete_All_Nodes()
{
    node *ptr,*preptr;
    ptr=head;
    preptr=NULL;
    if(ptr==NULL)
    {
        printf("\nThe list is Empty\n");
        return;
    }
    do
    {
        preptr=ptr;
        ptr=ptr->next;
        head=ptr;
        free(preptr);
    }while(ptr!=NULL);

    printf("\nAll nodes are deleted\n");
    printf("\n_____________________________________________________________________________\n");
}

void Insert_At_Pos(int info,int pos)
{
    node *ptr,*new_node;
    ptr=head;
    new_node=(node *)malloc(sizeof(node));
    if(new_node==NULL)
    {
        printf("Overflow Condition");
        return;
    }
    int i;
    new_node->data=info;
    new_node->next=NULL;
    if(pos==1)
    {
        new_node->next=head;
        head=new_node;
    }
    else if(pos>0)
    {
        for(i=1;i<pos-1 && ptr!=NULL ;i++)
            ptr=ptr->next;
        if(ptr)
        {
            new_node->next=ptr->next;
            ptr->next=new_node;
        }
    }
    else
        printf("\nInvalid Position\n");

    printf("\n____________________________________________________________________________\n");
}

void reverse()
{
    node *p,*q,*r;          //p : preptr , q : ptr , r : postptr
    p=q=NULL;
    r=head;
    while(r!=NULL)
    {
        p=q;
        q=r;
        r=r->next;
        q->next=p;
    }
    head=q;

    printf("\nLinked list reversed\n");
    printf("\n____________________________________________________________________________\n");
}
```