Problem 1: Reverse a Linked List
Write a C program to reverse a singly linked list. The program should traverse the list, reverse the pointers between the nodes, and display the reversed list.
Requirements:
Define a function to reverse the linked list iteratively.
Update the head pointer to the new first node.
Display the reversed list.

Example Input:
rust
Copy code
Initial list: 10 -> 20 -> 30 -> 40
Example Output:
rust
Copy code
Reversed list: 40 -> 30 -> 20 -> 10

```
void reverse(Node** nodes)

{
    Node* prevnode = NULL, *nextnode=NULL;
    Node* cur = *nodes;
    while (cur!=NULL)
    {
        nextnode = cur->next;
        cur->next = prevnode;
        prevnode = cur;
        cur= nextnode;
    }
    *nodes = prevnode;
}
```

==========

Problem 2: Find the Middle Node
Write a C program to find and display the middle node of a singly linked list. If the list has an even number of nodes, display the first middle node.
Requirements:
Use two pointers: one moving one step and the other moving two steps.
When the faster pointer reaches the end, the slower pointer will point to the middle node.
Example Input:
rust
Copy code
List: 10 -> 20 -> 30 -> 40 -> 50
Example Output:
scss

Copy code
Middle node: 30

```
void MiddleNode(Node* mid)

{
    if (mid == NULL)
    {
        printf("The list is empty.\n");
        return;
    }
    int length = 0;
    Node* temp = mid;
    while (temp != NULL)
    {
        length++;
        temp = temp->next;
    }
    int middleIndex = length / 2;
    temp = mid;
    for (int i = 0; i < middleIndex; i++)
    {
        temp = temp->next;
    }
    printf("Middle node: %d\n", temp->data);
}
```

Create a linked list to store the name, class, section, roll number and marks of any 3 subjects for 5 students.

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node
{
    char name[10];
    int roll;
    int class;
    char section;
    int marks[3];
    struct node *next;
}node;
```

```c
node *createnode(void);
int main()
{
    node *first=createnode();
    first->next=createnode();
    //first->next->next=createnode();
    //first->next->next->next=createnode();
    //first->next->next->next->next=createnode();
    node *temp;
    temp=first;
    while(temp!=NULL)
    {
        printf("name= %s \n",temp->name);
        printf("roll= %d \n",temp->roll);
        printf("class= %d \n",temp->class);
        printf("section= %c \n",temp->section);
        for(int i=0;i<3;i++)
        {
        printf("marks= %d ",temp->marks[i]);
        }
        printf("\n");
        temp=temp->next;
    }
    return 0;
}
node * createnode(void)
{
    node *newnode=(node *)malloc(sizeof(node));
    printf("enter the name \n");
    scanf("%s",newnode->name);
    printf("enter the roll number \n");
    scanf("%d",&newnode->roll);
    printf("enter the class \n");
    scanf("%d",&newnode->class);
    printf("enter the section \n");
    getchar();
    scanf("%c",&newnode->section);
    printf("enter the marks \n");
    for(int i=0;i<3;i++)
    {
        scanf("%d",&newnode->marks[i]);
    }
    newnode->next=NULL;
    return newnode;
}
```