create a node in a linked list which will have the following details of student   1. Name, roll number, class, section, an array having marks of any three subjects Create a liked for 5 students and print it.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


typedef struct Student {

   char name[20];

   int roll_no;

   int class;

   char section;

   int marks[3];

   struct Student *next;

} Student;


Student* createStudent(char name[], int roll_no, int class, char section, int marks[]) {

   Student *newStudent = (Student *)malloc(sizeof(Student));

   strcpy(newStudent->name, name);

   newStudent->roll_no = roll_no;

   newStudent->class = class;

   newStudent->section = section;

   for (int i = 0; i < 3; i++) {

      newStudent->marks[i] = marks[i];

   }

   newStudent->next = NULL;

   return newStudent;

}


int main() {

   Student *first = NULL;

   Student *temp = NULL;
```

```
char name[20];
int roll_no, class;
char section;
int marks[3];

for (int i = 0; i < 5; i++) {
    printf("Enter details for Student %d:\n", i + 1);

    printf("Enter name: ");
    scanf("%s", name);

    printf("Enter roll number: ");
    scanf("%d", &roll_no);

    printf("Enter class: ");
    scanf("%d", &class);

    printf("Enter section: ");
    scanf(" %c", &section);  // Note the space before %c to consume any leftover newline

    printf("Enter marks for 3 subjects: ");
    for (int j = 0; j < 3; j++) {
        scanf("%d", &marks[j]);
    }

    if (first == NULL) {
        first = createStudent(name, roll_no, class, section, marks);
        temp = first;
    } else {
        temp->next = createStudent(name, roll_no, class, section, marks);
        temp = temp->next;
        }
    }

    temp = first;
    printf("\nStudent Details:\n");
    while (temp != NULL) {
        printf("Name: %s\n", temp->name);
        printf("Roll Number: %d\n", temp->roll_no);
        printf("Class: %d\n", temp->class);
        printf("Section: %c\n", temp->section);
        printf("Marks: %d, %d, %d\n", temp->marks[0], temp->marks[1], temp->marks[2]);

        temp = temp->next;
    }

    return 0;
}
```

Output:

```
Enter details for Student 1:
Enter name: Anu
Enter roll number: 2
Enter class: 10
Enter section: A
Enter marks for 3 subjects: 76 79 80
Enter details for Student 2:
Enter name: Arun
Enter roll number: 9
Enter class: 10
Enter section: B
Enter marks for 3 subjects: 88 89 91
Enter details for Student 3:
```

```
Enter name: Akhil

Enter roll number: 9

Enter class: 9

Enter section: C

Enter marks for 3 subjects: 78 89 90

Enter details for Student 4:

Enter name: Sunitha

Enter roll number: 20

Enter class: 11

Enter section: A

Enter marks for 3 subjects: 88 87 99

Enter details for Student 5:

Enter name: Alicia

Enter roll number: 12

Enter class: 11

Enter section: C

Enter marks for 3 subjects: 96 94 91


Student Details:

Name: Anu

Roll Number: 2

Class: 10

Section: A

Marks: 76, 79, 80

Name: Arun

Roll Number: 9

Class: 10

Section: B

Marks: 88, 89, 91

Name: Akhil

Roll Number: 9
```

Class: 9

Section: C

Marks: 78, 89, 90

Name: Sunitha

Roll Number: 20

Class: 11

Section: A

Marks: 88, 87, 99

Name: Alicia

Roll Number: 12

Class: 11

Section: C

Marks: 96, 94, 91

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:

1. Define a function to reverse the linked list iteratively.
2. Update the head pointer to the new first node.
3. 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

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node{
   int data;
   struct node* next;
}Node;
void InsertEnd(Node**,int);
void printList(Node*);
void reverseList(Node**);
int main()
{
   Node* head=NULL;
   int n,value;
   printf("Enter the number of nodes:");
   scanf("%d",&n);
   for(int i=0;i<n;i++)
   {
      printf("Enter value for node %d:",i+1);
      scanf("%d",&value);
      InsertEnd(&head,value);
   }
   printf("Initial list:\n");
   printList(head);
   reverseList(&head);
   printf("Reversed list:");
   printList(head);
   return 0;
}
void InsertEnd(Node** ptrHead,int data)
{
   Node* new_node=(Node*)malloc(sizeof(Node));
   new_node->data=data;
   new_node->next=NULL;
   if(*ptrHead==NULL)
   {
      *ptrHead=new_node;
      return;
   }
   Node* ptrTail=*ptrHead;
   while(ptrTail->next!=NULL)
   {
      ptrTail=ptrTail->next;
   }
   ptrTail->next=new_node;
}
void printList(Node* head)
{
   Node* temp=head;
   while(temp!=NULL)
   {
      printf("%d -> ",temp->data);
      temp=temp->next;
   }
}
void reverseList(Node** head)
{
   Node *prev=NULL;
   Node *current=*head;
   Node *next=NULL;
   while(current!=NULL)
   {
      next=current->next;
      current->next=prev;
      prev=current;
      current=next;
   }
   *head=prev;
}
```

Output:

```
Enter the number of nodes:5
Enter value for node 1:10
Enter value for node 2:203
Enter value for node 3:0
Enter value for node 4:40
Enter value for node 5:50
Initial list:10 -> 20 -> 0 -> 40 -> 50 ->
Reversed list:50 -> 40 -> 0 -> 20 -> 10 ->
```

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:

1. Use two pointers: one moving one step and the other moving two steps.
2. 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
```

```
 #include<stdio.h>
#include<stdlib.h>
typedef struct node{
   int data;
   struct node* next;
}Node;
void insertEnd(Node**,int);
void printList(Node*);
void findMiddle(Node*);
int main()
{
   Node* head=NULL;
   int n,value;
   printf("Enter the number of nodes:");
   scanf("%d",&n);
   for(int i=0;i<n;i++)
   {
      printf("Enter the value for node %d:",i+1);
      scanf("%d",&value);
      insertEnd(&head,value);
   }
   printf("List:\n");
   printList(head);
   findMiddle(head);
   return 0;
}
void insertEnd(Node** ptrHead,int data)
{
   Node* new_node=(Node*)malloc(sizeof(Node));
   new_node->data=data;
   new_node->next=NULL;
   if(*ptrHead==NULL)
   {
      *ptrHead=new_node;
      return;
   }
   Node *ptrTail=*ptrHead;
   while(ptrTail->next!=NULL)
   {
      ptrTail=ptrTail->next;
   }
   ptrTail->next=new_node;
}
void printList(Node* head)
{
   Node* temp=head;
   while(temp!=NULL)
   {
      printf("%d -> ",temp->data);
      temp=temp->next;
   }
}
void findMiddle(Node* head)
{
   if(head==NULL)
   {
      printf("List is empty");
      return;
   }
   Node* slow=head;

   Node* fast=head;

   while(fast!=NULL && fast->next!=NULL)

   {

      slow=slow->next;

      fast=fast->next->next;

   }

   printf("Middle node:%d\n",slow->data);

}
```

Output:

```
Enter the number of nodes:5

Enter the value for node 1:10

Enter the value for node 2:15

Enter the value for node 3:20

Enter the value for node 4:25

Enter the value for node 5:30

List:10 -> 15 -> 20 -> 25 -> 30 ->

Middle node:20
```

Problem 3: Detect and Remove a Cycle in a Linked List

Write a C program to detect if a cycle (loop) exists in a singly linked list and remove it if present. Use Floyd's Cycle Detection Algorithm (slow and fast pointers) to detect the cycle.

Requirements:

1. Detect the cycle in the list.
2. If a cycle exists, find the starting node of the cycle and break the loop.
3. Display the updated list.

Example Input:

rust

Copy code

List: 10 -> 20 -> 30 -> 40 -> 50 -> (points back to 30)

Example Output:

rust

Copy code

```
Cycle detected and removed.
Updated list: 10 -> 20 -> 30 -> 40 -> 50
```

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

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

void insertEnd(Node** head, int data);
int detectAndRemoveCycle(Node* head);
void printList(Node* head);

int main() {
    Node* head = NULL;
    insertEnd(&head, 10);
    insertEnd(&head, 20);
    insertEnd(&head, 30);
    insertEnd(&head, 40);
    insertEnd(&head, 50);

    head->next->next->next->next->next = head->next->next;

    if (detectAndRemoveCycle(head)) {
        printf("Cycle detected and removed.\n");
    } else {
        printf("No cycle detected.\n");
    }

    printList(head);
    return 0;
}

void insertEnd(Node** head, int data) {
    Node* new_node = (Node*)malloc(sizeof(Node));
    new_node->data = data;
    new_node->next = NULL;
    if (*head == NULL) {
        *head = new_node;
        return;
    }
    Node* temp = *head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = new_node;
}

int detectAndRemoveCycle(Node* head) {
    Node *slow = head, *fast = head;
    while (fast != NULL && fast->next != NULL) {
        slow = slow->next;
        fast = fast->next->next;
        if (slow == fast) {
            Node* start = head;
            if (slow == head) {
                while (fast->next != slow) {
```

```
                fast = fast->next;
            }
            fast->next = NULL;
        } else {
            while (start->next != slow->next) {
                start = start->next;
                slow = slow->next;
            }
            slow->next = NULL;
        }
        return 1;
    }
  }
  return 0;
}

void printList(Node* head) {
  while (head != NULL) {
    printf("%d -> ", head->data);
    head = head->next;
  }
}
```

Output:

```
Cycle detected and removed.
10 -> 20 -> 30 -> 40 -> 50 ->
```