1) 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 linked list for 5 students and print it.

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

typedef struct student {
    char name[50];
    int rollNumber;
    int class;
    char section[5];
    int marks[3];
    struct student *next;
} Student;

Student *createNode() {
    Student *newNode = (Student *)malloc(sizeof(Student));
    if (!newNode) {
        printf("Memory allocation failed!\n");
        exit(1);
    }

    printf("Enter name: ");
    scanf(" %[^\n]", newNode->name);

    printf("Enter roll number: ");
    scanf("%d", &newNode->rollNumber);

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

    printf("Enter section: ");
    scanf(" %s", newNode->section);

    printf("Enter marks of three subjects: ");
    for (int i = 0; i < 3; i++) {
        scanf("%d", &newNode->marks[i]);
    }

    newNode->next = NULL;
    return newNode;
}

void printList(Student *head) {
```

```
    Student *temp = head;
    printf("\n-------------------------------\n");
    printf("Student Details\n");
    printf("-------------------------------\n");
    printf("%-15s %-10s %-10s %-10s %-20s\n", "Name", "Roll No.", "Class", "Section",
"Marks");
    printf("-----------------------------------------------------------\n");

    while (temp != NULL) {
        printf("%-15s %-10d %-10d %-10s ", temp->name, temp->rollNumber, temp->class,
temp->section);
        for (int i = 0; i < 3; i++) {
            printf("%d ", temp->marks[i]);
        }
        printf("\n");
        temp = temp->next;
    }
}

int main() {
    Student *first = NULL, *temp;
    int numStudents = 5;

    for (int i = 0; i < numStudents; i++) {
        printf("\nEnter details for student %d:\n", i + 1);
        Student *newNode = createNode();

        if (first == NULL) {
            first = newNode;
        } else {
            temp = first;
            while (temp->next != NULL) {
                temp = temp->next;
            }
            temp->next = newNode;
        }
    }

    printList(first);

    temp = first;
    while (temp != NULL) {
        Student *next = temp->next;
        free(temp);
        temp = next;
    }

    return 0;
```

}

i/p

```
Enter details for student 1:
Enter name: John
Enter roll number: 1
Enter class: 10
Enter section: A
Enter marks of three subjects: 85 90 88

Enter details for student 2:
Enter name: Alice
Enter roll number: 2
Enter class: 10
Enter section: B
Enter marks of three subjects: 78 82 80

Enter details for student 3:
Enter name: Bob
Enter roll number: 3
Enter class: 10
Enter section: C
Enter marks of three subjects: 92 88 84

Enter details for student 4:
Enter name: Clara
Enter roll number: 4
Enter class: 11
Enter section: A
Enter marks of three subjects: 89 91 85

Enter details for student 5:
Enter name: Daniel
Enter roll number: 5
Enter class: 11
Enter section: B
Enter marks of three subjects: 76 80 79
```

o/p

```
------------------------------
Student Details
------------------------------
Name       Roll No.   Class      Section     Marks
------------------------------------------------------------
John          1             10         A          85 90 88
Alice         2             10         B          78 82 80
```

Bob	3	10	C	92 88 84
Clara	4	11	A	89 91 85
Daniel	5	11	B	76 80 79

2) Implementation of adding nodes to a linked list

```
#include <stdio.h>

#include <stdlib.h>

typedef struct node {

    int data;
    struct node *next;

}Node;

void InsertFront (Node **,  int);

void InsertMiddle (Node *,  int,  int);

void Insert End(Node **,  int);

void printList(Node *);

int main() {

Node *head = NULL;

InsertEnd(&head, 6);

InsertEnd(&head, 8);

InsertEnd(&head, 10);

InsertFront(&head, 4);

InsertFront(&head, 0);

InsertMiddle (head, 2, 7);

printList(head);

return 0;

}
```

```
void InsertEnd (Node **ptrHead, int nData) {
Node *newNode = (Node *)malloc(sizeof(Node));
if (newNode == NULL) {
printf("Memory allocation failed.\n");
return;
}
newNode->data = nData;
newNode->next = NULL;
if (*ptrHead == NULL) {
*ptrHead = newNode;
} else {
Node *ptrTail = *ptrHead;
while (ptrTail->next != NULL) {
ptrTail = ptrTail->next;
}
ptrTail->next = newNode;
}
}
void InsertFront (Node  **ptrHead, int nData) {
Node  *newNode =  (Node *)malloc(sizeof(Node));
if (newNode == NULL) {
printf("Memory allocation failed.\n");
return;
}
```

```
newNode->data =  nData;
newNode->next = *ptrHead;
*ptrHead = newNode;
}
void InsertMiddle (Node *ptrHead, int after, int nData) {
if (ptrHead == NULL) {
printf("The list is empty. Cannot insert at position %d.\n", after);
return;
}
Node *newNode = (Node *)malloc(sizeof(Node));
if (newNode == NULL) {
printf("Memory allocation failed.\n");
return;
}
newNode->data = nData;
newNode->next = NULL;
Node *ptrCurrent = ptrHead;
int count = 1;
while (ptrCurrent != NULL && count < after) {
ptrCurrent =  ptrCurrent->next;
count++;
}
if (ptrCurrent == NULL) {
```

```
printf("Invalid position: List has fewer than %d nodes.\n", after);
free (newNode);
return;

}

newNode->next = ptrCurrent -> next;

ptrCurrent->next = newNode;

}

void printList(Node *node) {

while (node != NULL) {

printf("%d -> ", node->data);

Node = node->next;

}
printf("NULL\n");
}
```

o/p

0 -> 4 -> 7 -> 6 -> 8 -> 10 -> NULL

3. 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 -> 38 -> 20 -> 10

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

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

Node *createNode(int data) {
    Node *newNode = (Node *)malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory allocation failed!\n");
        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

void appendNode(Node **head, int data) {
    Node *newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
    } else {
        Node *temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

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;
}

void displayList(Node *head) {
    Node *temp = head;
    while (temp != NULL) {
        printf("%d", temp->data);
        if (temp->next != NULL) {
            printf(" -> ");
        }
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    Node *head = NULL;

    appendNode(&head, 10);
    appendNode(&head, 20);
    appendNode(&head, 30);
    appendNode(&head, 40);

    printf("Initial list: ");
    displayList(head);

    reverseList(&head);

    printf("Reversed list: ");
    displayList(head);

    Node *temp = head;
    while (temp != NULL) {
        Node *next = temp->next;
        free(temp);
        temp = next;
    }

    return 0;
}
```

4) 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

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

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

void InsertEnd(Node **head, int data) {
    Node *newNode = (Node *)malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory allocation failed!\n");
        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;

    if (*head == NULL) {
        *head = newNode;
```

```
    } else {
        Node *temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

void findMiddle(Node *head) {
    if (head == NULL) {
        printf("The list is empty.\n");
        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);
}

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

int main() {
    Node *head = NULL;

    InsertEnd(&head, 10);
    InsertEnd(&head, 20);
    InsertEnd(&head, 30);
    InsertEnd(&head, 40);
    InsertEnd(&head, 50);

    printf("List: ");
    printList(head);
```

```
    findMiddle(head);

    Node *temp = head;
    while (temp != NULL) {
        Node *next = temp->next;
        free(temp);
        temp = next;
    }

    return 0;
}
```

o/p

```
List: 10 -> 20 -> 30 -> 40 -> 50
Middle node: 30
```

5) 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:

Detect the cycle in the list.

If a cycle exists, find the starting node of the cycle and break the loop.

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;

Node *createNode(int data) {
    Node *newNode = (Node *)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

void appendNode(Node **head, int data) {
    Node *newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
    } else {
        Node *temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

void createCycle(Node *head, int index) {
    if (head == NULL || index < 0) return;

    Node *targetNode = NULL, *temp = head;
    int counter = 0;

    while (temp->next != NULL) {
        if (counter == index) {
            targetNode = temp;
        }
        temp = temp->next;
        counter++;
    }

    if (targetNode) {
        temp->next = targetNode;
        printf("Cycle created: last node points to node with data %d.\n", targetNode->data);
    }
}
```

```
void detectAndRemoveCycle(Node *head) {
    Node *slow = head, *fast = head;

    while (fast && fast->next) {
        slow = slow->next;
        fast = fast->next->next;
        if (slow == fast) break;
    }

    if (!fast || !fast->next) {
        printf("No cycle detected.\n");
        return;
    }

    printf("Cycle detected and removed.\n");

    slow = head;
    while (slow != fast->next) {
        slow = slow->next;
        fast = fast->next;
    }

    fast->next = NULL;
}

void displayList(Node *head) {
    while (head) {
        printf("%d", head->data);
        if (head->next) {
            printf(" -> ");
        }
        head = head->next;
    }
    printf("\n");
}

int main() {
    Node *head = NULL;
    int n, data, cycleIndex;

    printf("Enter the number of elements in the linked list: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &data);
        appendNode(&head, data);
    }
```

```
    printf("\nInitial list: ");
    displayList(head);

    printf("Do you want to create a cycle? (Enter -1 for no cycle or index of node [0-%d] to
point the last node): ", n - 1);
    scanf("%d", &cycleIndex);

    if (cycleIndex != -1) {
        createCycle(head, cycleIndex);
    }

    detectAndRemoveCycle(head);

    printf("Updated list: ");
    displayList(head);

    return 0;
}
```

i/p

```
Enter the number of elements in the linked list: 5
Enter element 1: 10
Enter element 2: 20
Enter element 3: 30
Enter element 4: 40
Enter element 5: 50
Do you want to create a cycle? (Enter -1 for no cycle or index of node [0-4] to point the last
node): 2
```

o/p

```
Initial list: 10 -> 20 -> 30 -> 40 -> 50
Cycle created: last node points to node with data 30.
Cycle detected and removed.
Updated list: 10 -> 20 -> 30 -> 40 -> 50
```