1.create a node in a linked list which will have the following details of student 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>

struct Student {
   char name[50];
   int roll_no;
   char class_name[20];
   char section;
   int marks[3]; // Marks for 3 subjects
   struct Student* next;
};

struct Student* createStudent(char name[], int roll_no, char class_name[], char section, int marks[3]) {
   struct Student* newStudent = (struct Student*)malloc(sizeof(struct Student));
   strcpy(newStudent->name, name);
   newStudent->roll_no = roll_no;
   strcpy(newStudent->class_name, class_name);
   newStudent->section = section;
   for (int i = 0; i < 3; i++) {
      newStudent->marks[i] = marks[i];
   }
   newStudent->next = NULL;
   return newStudent;
```

```
}

void printList(struct Student* head) {
    while (head != NULL) {
        printf("Name: %s, Roll No: %d, Class: %s, Section: %c, Marks: %d, %d, %d\n",
               head->name, head->roll_no, head->class_name, head->section,
               head->marks[0], head->marks[1], head->marks[2]);
        head = head->next;
    }
}

int main() {
    int marks1[] = {85, 90, 88};
    int marks2[] = {75, 80, 78};
    int marks3[] = {95, 92, 98};
    int marks4[] = {60, 65, 63};
    int marks5[] = {70, 72, 68};

    struct Student* head = createStudent("John", 1, "10th", 'A', marks1);
    head->next = createStudent("Jane", 2, "10th", 'B', marks2);
    head->next->next = createStudent("Mike", 3, "10th", 'A', marks3);
    head->next->next->next = createStudent("Anna", 4, "10th", 'B', marks4);
    head->next->next->next->next = createStudent("Chris", 5, "10th", 'A', marks5);

    printf("Student List:\n");
    printList(head);

    return 0;
```

```
}
```

2. 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

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

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

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

void reverseList(struct Node** head) {
  struct Node *prev = NULL, *current = *head, *next = NULL;
  while (current != NULL) {
    next = current->next;
    current->next = prev;
    prev = current;
    current = next;
  }
  *head = prev;
}

void printList(struct Node* head) {
  while (head != NULL) {
```

```
    printf("%d -> ", head->data);
    head = head->next;
  }
  printf("NULL\n");
}

int main() {
  struct Node* head = createNode(10);
  head->next = createNode(20);
  head->next->next = createNode(30);
  head->next->next->next = createNode(40);

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

  reverseList(&head);
  printf("Reversed list: ");
  printList(head);

  return 0;
}
```

3. 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>

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

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

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

int main() {
   struct Node* head = createNode(10);
   head->next = createNode(20);
   head->next->next = createNode(30);
   head->next->next->next = createNode(40);
   head->next->next->next->next = createNode(50);

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

   findMiddle(head);

   return 0;
}
```

4. 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
has context menu

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

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

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

void detectAndRemoveCycle(struct Node* head) {
  struct Node* slow = head;
  struct Node* fast = head;
  struct Node* start = head;

  while (fast != NULL && fast->next != NULL) {
    slow = slow->next;
    fast = fast->next->next;

    if (slow == fast) {
      while (start != slow) {
        start = start->next;
        slow = slow->next;
```

```
        }
        struct Node* cycleStart = start;
        struct Node* prev = NULL;

        while (slow->next != cycleStart) {
          slow = slow->next;
        }
        slow->next = NULL;

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

    printf("No cycle detected.\n");
}

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

int main() {
    struct Node* head = createNode(10);
    head->next = createNode(20);
    head->next->next = createNode(30);
    head->next->next->next = createNode(40);
    head->next->next->next->next = createNode(50);

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

    detectAndRemoveCycle(head);

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

    return 0;
}
```