1.Find Loop in Linked List.

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

struct Node
{
    int data;
    struct Node *next;
};
int isloop(struct Node*);
int main()
{
  struct Node *head,*n1,*n2;
  head=(struct Node *)malloc(sizeof(struct Node));
  n1=(struct Node *)malloc(sizeof(struct Node));
  n2=(struct Node *)malloc(sizeof(struct Node));


  head->data=10;
  head->next=n1;
  n1->data=20;
  n1->next=n2;
  n2->data=40;
  n2->next=NULL;

  int i=isloop(head);
  printf("%d",i);
  if(i==1){
    printf("\nNO LOOP!!!");
  }
  return 0;
}
int isloop(struct Node *p){
    struct Node *temp=p,*mov;
    while(temp->next!=NULL){
        mov=temp->next;
        while(mov->next!=NULL){
            if(temp->next==mov->next){
                printf("\nLOOP!!!");
                return 0;
            }
            mov=mov->next;
        }
```

```
        temp=temp->next;
    }
    return 1;
}
```

2.Create two linked list in one linked {1,2,3,4}and in the 2nd linked list will have value{7,8,9}.Concatenate both the linked list and display the concatenated linked list.

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

struct Node
{
    int data;
    struct Node *next;
};
struct Node* concat(struct Node*,struct Node*);
void display(struct Node*);
int main()
{
  struct Node *head,*n1,*n2,*shead,*s1,*s2;
  head=(struct Node *)malloc(sizeof(struct Node));
  n1=(struct Node *)malloc(sizeof(struct Node));
  n2=(struct Node *)malloc(sizeof(struct Node));
  shead=(struct Node *)malloc(sizeof(struct Node));
  s1=(struct Node *)malloc(sizeof(struct Node));
  s2=(struct Node *)malloc(sizeof(struct Node));


  head->data=10;
  head->next=n1;
  n1->data=20;
  n1->next=n2;
  n2->data=30;
  n2->next=NULL;
  shead->data=40;
  shead->next=s1;
  s1->data=50;
  s1->next=s2;
  s2->data=60;
  s2->next=NULL;

  printf("\nFirst LIST::\n");
```

```
  display(head);
  printf("\nSecond LIST::\n");
  display(shead);

  head=concat(head,shead);
  printf("\nConcated LIST::\n");
  display(head);

  return 0;
}
void display(struct Node *p)
{
    while( p != NULL)
    {
        printf("%d->",p->data);
        p=p->next;
    }
}
struct Node* concat(struct Node* head,struct Node* shead){
    struct Node* temp=head;
    while(temp->next!=NULL){
        temp=temp->next;
    }
    temp->next=shead;
    return(head);

}
```

3.Problem Statement: Automotive Manufacturing Plant Management System

Objective:

Develop a program to manage an **automotive manufacturing plant's operations** using a **linked list** in C programming. The system will allow creation, insertion, deletion, and searching operations for managing assembly lines and their details.

Requirements

Data Representation

1. **Node Structure:**
   Each node in the linked list represents an assembly line.
   Fields:
   - lineID (integer): Unique identifier for the assembly line.
   - lineName (string): Name of the assembly line (e.g., "Chassis Assembly").

- capacity (integer): Maximum production capacity of the line per shift.
- status (string): Current status of the line (e.g., "Active", "Under Maintenance").
- next (pointer to the next node): Link to the next assembly line in the list.

2. **Linked List:**
   - The linked list will store a dynamic number of assembly lines, allowing for additions and removals as needed.

Features to Implement

1. **Creation:**
   - Initialize the linked list with a specified number of assembly lines.
2. **Insertion:**
   - Add a new assembly line to the list either at the beginning, end, or at a specific position.
3. **Deletion:**
   - Remove an assembly line from the list by its lineID or position.
4. **Searching:**
   - Search for an assembly line by lineID or lineName and display its details.
5. **Display:**
   - Display all assembly lines in the list along with their details.
6. **Update Status:**
   - Update the status of an assembly line (e.g., from "Active" to "Under Maintenance").

Example Program Flow

1. **Menu Options:**
   Provide a menu-driven interface with the following operations:
   - Create Linked List of Assembly Lines
   - Insert New Assembly Line
   - Delete Assembly Line
   - Search for Assembly Line
   - Update Assembly Line Status
   - Display All Assembly Lines
   - Exit
2. **Sample Input/Output:**
   Input:
   - Number of lines: 3
   - Line 1: ID = 101, Name = "Chassis Assembly", Capacity = 50, Status = "Active".
   - Line 2: ID = 102, Name = "Engine Assembly", Capacity = 40, Status = "Under Maintenance".

Output:

- Assembly Lines:
  - Line 101: Chassis Assembly, Capacity: 50, Status: Active
  - Line 102: Engine Assembly, Capacity: 40, Status: Under Maintenance

Linked List Node Structure in C

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


// Structure for a linked list node

typedef struct AssemblyLine {

  int lineID;                  // Unique line ID

  char lineName[50];           // Name of the assembly line

  int capacity;                // Production capacity per shift

  char status[20];             // Current status of the line

  struct AssemblyLine* next;   // Pointer to the next node

} AssemblyLine;
```

Operations Implementation

1. Create Linked List

- Allocate memory dynamically for AssemblyLine nodes.
- Initialize each node with details such as lineID, lineName, capacity, and status.

2. Insert New Assembly Line

- Dynamically allocate a new node and insert it at the desired position in the list.

3. Delete Assembly Line

- Locate the node to delete by lineID or position and adjust the next pointers of adjacent nodes.

4. Search for Assembly Line

- Traverse the list to find a node by its lineID or lineName and display its details.

5. Update Assembly Line Status

- Locate the node by lineID and update its status field.

6. Display All Assembly Lines

- Traverse the list and print the details of each node.

Sample Menu

Menu:

1. Create Linked List of Assembly Lines

2. Insert New Assembly Line

3. Delete Assembly Line

4. Search for Assembly Line

5. Update Assembly Line Status

6. Display All Assembly Lines

7. Exit

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
typedef struct AssemblyLine {
   int lineID;
   char lineName[50];
   int capacity;
   char status[20];
   struct AssemblyLine* next;
}AssemblyLine;
AssemblyLine* create(AssemblyLine*);
AssemblyLine* insert(AssemblyLine*);
AssemblyLine* delete(AssemblyLine*);
AssemblyLine* update(AssemblyLine*);
void search(AssemblyLine*);
void display(AssemblyLine*);
void sdisplay(AssemblyLine*);
int main(){
    int op;
    printf("\nASSEMBLY LINE");
    AssemblyLine* head=NULL;
    while(1){
```

```
        printf("\nMenu");
        printf("\n1. Create Linked List of Assembly Lines");
        printf("\n2. Insert New Assembly Line");
        printf("\n3. Delete Assembly Line");
        printf("\n4. Search for Assembly Line");
        printf("\n5. Update Assembly Line Status");
        printf("\n6. Display All Assembly Lines");
        printf("\n7. Exit");
        printf("\nChoose Option::");
        scanf("%d",&op);
        switch(op){
            case 1:
                head=create(head);
                break;
            case 2:
                head=insert(head);
                break;
            case 3:
                head=delete(head);
                break;
            case 4:
                search(head);
                break;
            case 5:
                head=update(head);
                break;
            case 6:
                display(head);
                break;
            case 7:
                printf("\nExiting....");
                free(head);
                return 0;
        }
    }
}
AssemblyLine* create(AssemblyLine* head){
    AssemblyLine *temp,*newnode;
    newnode=(AssemblyLine*)malloc(sizeof(AssemblyLine));
    printf("\nEnter LineID::");
    scanf("%d",&newnode->lineID);
    printf("\nEnter Line Name::");
    scanf(" %[^\n]",newnode->lineName);
```

```
    printf("\nEnter Line Capacity::");
    scanf("%d",&newnode->capacity);
    printf("\nEnter Status(ACTIVE or UNDERMAINTAINANCE)::");
    scanf(" %[^\n]",newnode->status);
    newnode->next=NULL;
    if(head==NULL){
        head=newnode;
    }
    else{
        temp=head;
        while(temp->next!=NULL){
            temp=temp->next;
        }
        temp->next=newnode;
    }
    return head;
}
AssemblyLine* insert(AssemblyLine* head){
    int pos;
    AssemblyLine *temp=head,*newnode;
    printf("\nEnter position to add new::");
    scanf("%d",&pos);
    for(int i=1;i<pos-1;i++){
        temp=temp->next;
    }
    newnode=(AssemblyLine*)malloc(sizeof(AssemblyLine));
    printf("\nEnter LineID::");
    scanf("%d",&newnode->lineID);
    printf("\nEnter Line Name::");
    scanf(" %[^\n]",newnode->lineName);
    printf("\nEnter Line Capacity::");
    scanf("%d",&newnode->capacity);
    printf("\nEnter Status(ACTIVE or UNDERMAINTAINANCE)::");
    scanf(" %[^\n]",newnode->status);
    if(pos==1){
        newnode->next=head;
        head=newnode;
    }
    else{
        newnode->next=temp->next;
        temp->next=newnode;
    }
    return head;
```

```
}
AssemblyLine* delete(AssemblyLine *head){
    int pos;
    AssemblyLine *temp=head,*prev;
    printf("\nEnter position of Assembly line to delete::");
    scanf("%d",&pos);
    if(pos==1){
        head=head->next;
    }
    else{
        for(int i=0;i<pos-1;i++){
            prev=temp;
            temp=temp->next;
        }
        prev->next=temp->next;
    }
    free(temp);
    printf("\nAssembly Line Deleted..");
    return head;

}
void search(AssemblyLine *head){
    AssemblyLine *temp=head;
    int id;
    printf("\nEnter ID to search::");
    scanf("%d",&id);
    while(temp!=NULL){
        if(temp->lineID==id){
            sdisplay(temp);
        }
        else{
            printf("\nNOT FOUND");
        }
        temp=temp->next;
    }
}
AssemblyLine* update(AssemblyLine* head){
    int id;
    AssemblyLine *temp=head;
    printf("\nEnter ID of Assembly Line to Update::");
    scanf("%d",&id);
    while(temp!=NULL){
```

```
        if(temp->lineID==id){
            printf("\nEnter new Status(ACTIVE or
UNDERMAINTAINANCE)::");
            scanf(" %[^\n]",temp->status);
        }
        else{
            printf("\nNOT FOUND");
        }
        temp=temp->next;
    }
    return head;

}
void display(AssemblyLine* head){
    AssemblyLine *temp=head;
    while(temp!=NULL){
        printf("\nLINE %d, NAME::%s, CAPACITY::%d,
STATUS::%s->",temp->lineID,temp->lineName,temp->capacity,temp->status);
        temp=temp->next;
    }
    printf("\nNULL");
}
void sdisplay(AssemblyLine *temp){
    while(temp!=NULL){
        printf("\nLINE %d, NAME::%s, CAPACITY::%d,
STATUS::%s->",temp->lineID,temp->lineName,temp->capacity,temp->status);
        break;
    }
}
```