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

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

Example Program Flow
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
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
*/

#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;
void createNode(AssemblyLine **);
void insertNode(AssemblyLine **);
void printNode(AssemblyLine *);
void deleteNode(AssemblyLine**);
void searchNode(AssemblyLine* );
void update(AssemblyLine*);
int main()
{
    int choice;
    AssemblyLine *head=NULL;
    printf("Menu:\n1. Create Linked List of Assembly Lines\n2. Insert New
Assembly Line\n3. Delete Assembly Line\n4. Search for Assembly Line\n5. Update
Assembly Line Status\n6. Display All Assembly Lines\n7. Exit\n");
    while(1)
    {
        printf("enter your choice \n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                createNode(&head);
```

```
                break;

            case 2:
                insertNode(&head);
                break;

            case 3:
                deleteNode(&head);
                break;

            case 4:
                searchNode(head);
                break;

            case 5:
                update(head);
                break;

            case 6:
                printNode(head);
                break;

            case 7:
                printf("exiting the system \n");
                exit(0);

            default:
                printf("wrong choice\n");
        }
    }
    return 0;
}
void createNode(AssemblyLine** pointer)
{
    int n;
    printf("Enter the number of assembly lines: ");
    scanf("%d",&n);
    AssemblyLine* temp=NULL;
    for (int i = 0;i<n;i++)
    {
        AssemblyLine* newnode =(AssemblyLine*)malloc(sizeof(AssemblyLine));
        printf("line ID: ");
        scanf("%d", &newnode->lineID);
        printf("line Name: ");
        scanf("%s",newnode->lineName);
```

```
        printf("capacity: ");
        scanf("%d",&newnode->capacity);
        printf("status: ");
        scanf("%s",newnode->status);
        newnode->next=NULL;
        if (*pointer==NULL)
        {
            *pointer =newnode;
        } else
        {
            temp->next =newnode;
        }
        temp = newnode;
    }
}
void insertNode(AssemblyLine** pointer)
{
    AssemblyLine* newnode = (AssemblyLine*)malloc(sizeof(AssemblyLine));
    int index;
    printf("Enter the index position to insert the assembly line \n");
    scanf("%d", &index);
    printf("Line ID: \n");
    scanf("%d", &newnode->lineID);
    printf("Line Name: \n");
    scanf("%s", newnode->lineName);
    printf("Capacity: \n");
    scanf("%d", &newnode->capacity);
    printf("Status: \n");
    scanf("%s", newnode->status);
    newnode->next = NULL;
    if (index == 0)
    {
        newnode->next = *pointer;
        *pointer = newnode;
        return;
    }
    AssemblyLine* temp = *pointer;
    for (int i = 0; i < index - 1; i++)
    {
        if (temp == NULL)
        {
            printf("Invalid index \n");
            return;
        }
        temp = temp->next;
```

```
    }
    if (temp!=NULL)
    {
        newnode->next = temp->next;
        temp->next = newnode;
    }
}

void printNode(AssemblyLine * ptr)
{
    int i=1;
    while(ptr!=NULL)
    {
        printf("line %d details \n",i);
        printf("line ID: %d \n",ptr->lineID);
        printf("line Name: %s \n",ptr->lineName);
        printf("capacity: %d \n",ptr->capacity);
        printf("status: %s \n",ptr->status);
        printf("\n");
        ptr=ptr->next;
        i++;
    }
}
void deleteNode(AssemblyLine** pointer)
{
    int id;
    printf("Enter the Line ID to delete: ");
    scanf("%d", &id);
    AssemblyLine* temp = *pointer;
    AssemblyLine* prev = NULL;
    while (temp != NULL && temp->lineID != id)
    {
        prev = temp;
        temp = temp->next;
    }
    if (temp == NULL)
    {
        printf("Assembly line with ID %d not found.\n", id);
        return;
    }
    if (prev == NULL)
    {
        *pointer = temp->next;
    } else
    {
```

```
        prev->next = temp->next;
    }
    free(temp);
}
void searchNode(AssemblyLine* head)
{
    int choice, id;
    char name[50];
    printf("Search by:\n1. Line ID\n2. Line Name\n");
    scanf("%d", &choice);
    AssemblyLine* temp = head;
    if (choice == 1)
    {
        printf("Enter Line ID: ");
        scanf("%d", &id);
        while (temp != NULL)
        {
            if (temp->lineID == id)
            {
                printf("Line Found: %s, Capacity: %d, Status: %s\n",temp-
>lineName, temp->capacity, temp->status);
                return;
            }
            temp = temp->next;
        }
    } else if (choice == 2)
    {
        printf("Enter Line Name: ");
        scanf("%s", name);

        while (temp != NULL)
        {
            if (strcmp(temp->lineName, name) == 0)
            {
                printf("Line Found: ID: %d, Capacity: %d, Status: %s\n",
                       temp->lineID, temp->capacity, temp->status);
                return;
            }
            temp = temp->next;
        }
    }
}
void update(AssemblyLine* head)
{
    int id;
```

```
    char newStatus[20];
    printf("Enter Line ID to update status: ");
    scanf("%d", &id);
    printf("Enter new status: ");
    scanf("%s", newStatus);
    AssemblyLine* temp = head;
    while (temp != NULL)
    {
        if (temp->lineID == id)
        {
            strcpy(temp->status,newStatus);
            printf("Status updated successfully.\n");
            return;
        }
        temp = temp->next;
    }
    printf("Assembly line with ID %d not found.\n", id);
}
```

```
#include<stdio.h>
#include<stdlib.h>
struct stack
{
    int size;
    int top;
    int *s;
};
void create(struct stack *);
void push(struct stack *,int);
int pop(struct stack *);
void display(struct stack*);
int peek(struct stack*,int );
int main()
{
    int popped,val;
    struct stack st;
    create(&st);
    push(&st,10);
    push(&st,20);
    push(&st,30);
    push(&st,40);
    val=peek(&st,1);
    //popped=pop(&st);
    //printf("popped element is %d \n",popped);
```

```c
    printf("value at the position is %d \n",val);
    display(&st);
    return 0;
}
void create(struct stack *st)
{
    printf("enter size \n");
    scanf("%d",&st->size);
    st->top=-1;
    st->s=(int *)malloc(st->size*sizeof(int));
}
void push(struct stack *st,int x)
{
    if(st->top==st->size-1)
    {
        printf("stack overflow \n");
    }
    else
    {
        st->top++;
        st->s[st->top]=x;
    }
}
void display(struct stack *st)
{
    for(int i=st->top;i>=0;i--)
    {
        printf("%d",st->s[i]);
        printf("\n");
    }
}
int pop(struct stack *st)
{
    int x=0;
    if(st->top==-1)
    {
        printf("underflow \n");
    }
    else
    {
        x=st->s[st->top];
        st->top--;
    }
    return x;
}
```

```c
int peek(struct stack *st, int pos)
{
    int x=-1;
    if(st->top-pos+1<0)
    {
        printf("invalid position \n");
    }
    else
    {
        x=st->s[st->top-pos+1];
        return x;
    }
}
/*int n=-1,count=0;
    for (int i=st->top;i>=0;i--)
    {
        count++;
        if (count==position)
        {
            n=st->s[i];
            break;
        }
    }
    return n;*/
```