----------Insertion and deletion from LinkedList----------

*1.Inventory management system*

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<stdbool.h>
struct node{
   char material_name[100];
   struct node *next;
};

struct node* createNode(char* material_name){
   struct node *new_node=(struct node*)malloc(sizeof(struct node));
   strcpy((*new_node).material_name,material_name);
   new_node->next=NULL;
   return new_node;
}
struct node* insert(struct node *head,char *material_name){
   struct node *new_node=(struct node*)malloc(sizeof(struct node));
   if(head==NULL){
      head=new_node;
   }else{
      struct node* temp=head;
      while(temp->next!=NULL){
         temp=temp->next;
      }
      temp->next=new_node;
   }
   printf("Material '%s' added to the inventory.\n", material_name);
```

```
    return head;
}
struct node* delete(struct node *head,char *material_name){
    if(head==NULL){
        printf("The inventory is empty!");
        return head;
    }
    struct node *temp=head;
    struct node *prev=NULL;
    if(strcmp(material_name,temp->material_name)==0){
        head=temp->next;
        free(temp);
        printf("Material '%s' deleted from inventory.\n", material_name);
        return head;
    }
    while(temp->next!=NULL && strcmp(temp->material_name,material_name)!=0){
        temp=temp->next;
        prev=temp;
    }
    if (temp == NULL) {
        printf("Material '%s' not found in inventory.\n", material_name);
        return head;
    }
    prev->next=temp->next;
    free(temp);
    printf("Material '%s' deleted from inventory.\n", material_name);
        return head;
}
void display(struct node *head){
```

```
    if (head == NULL) {
        printf("Inventory is empty.\n");
        return;
    }
    struct node *temp=head;
    while(temp->next!=NULL){
        printf("- %s\n", temp->material_name);
        temp = temp->next;
    }
}

int main(){
    struct node *inventory=NULL;
    int choice;
    char material_name[30];
    bool is_on=true;
    while(is_on){
        printf("\nInventory Management System\n");
        printf("1. Insert a new raw material\n");
        printf("2. Delete a raw material\n");
        printf("3. Display current inventory\n");
        printf("4. Exit\n");
        printf("enter your choice :");
        scanf("%d",&choice);
        getchar();
        switch(choice){
            case 1:
                printf("Enter the name of the raw material to add: ");
                fgets(material_name, 100, stdin);
```

```
            material_name[strcspn(material_name, "\n")] = '\0';
            inventory=insert(inventory,material_name);
            break;
        case 2:
            printf("Enter the name of the raw material to add: ");
            fgets(material_name, 100, stdin);
            material_name[strcspn(material_name, "\n")] = '\0';
            inventory=delete(inventory,material_name);
            break;
        case 3:
            display(inventory);
            break;
        case 4:
            is_on=false;
            break;
        default:
            printf("Enter a valid option!");
      }
   }
   return 0;
}
```

*2.production line queue*

```
#include<stdio.h>
#include<stdbool.h>
#include<string.h>
#include<stdlib.h>
struct node{
  char name[30];
```

```
  struct node *next;
};

struct node* createnode(char *task_name){
    struct node *new_task=(struct node*)malloc(sizeof(struct node));
    strcpy(new_task->name,task_name);
    new_task->next=NULL;
    return new_task;
}
struct node* insertTask(struct node *head,char *task_name){
    struct node *new_task=createnode(task_name);
    if(head==NULL){
        return new_task;
    }
    struct node *temp=head;
    while(head->next!=NULL){
        temp=temp->next;
    }
    temp->next=new_task;
    return head;
}

struct node* deleteTask(struct node *head){
    if(head==NULL){
        printf("There are no tasks sheduled!");
        return head;
    }
    struct node *temp=head;
    head=head->next;
```

```
    free(temp);
    printf("Task completed and deleted from the queue.\n");
    return head;
}
void displayQueue(struct node *head){
    struct node *temp=head;
    while(temp!=NULL){
        printf("%s\n",temp->name);
        temp=temp->next;
    }
}
int main(){
    struct node *queue = NULL;
    int choice;
    char task_name[100];
    while (1) {
        printf("\n--- Production Line Task Queue ---\n");
        printf("1. Insert new task into the queue\n");
        printf("2. Delete completed task from the queue\n");
        printf("3. Display current task queue\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter the task name: ");
                getchar();
                fgets(task_name, sizeof(task_name), stdin);
                task_name[strcspn(task_name, "\n")] = '\0';
```

```
            queue = insertTask(queue, task_name);
            printf("Task '%s' added to the queue.\n", task_name);
            break;
        case 2:
            queue = deleteTask(queue);
            break;
        case 3:
            displayQueue(queue);
            break;
        case 4:
            printf("Exiting the program.\n");
            return 0;
        default:
            printf("Invalid choice. Please try again.\n");
      }
   }

   return 0;
}
```

3.

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<stdbool.h>
struct MaintenanceTask{
   char task_name[100]
   char task_date[20]
   struct MaintenanceTask *next;
```

```
};
struct MaintenanceTask* createTask(char *task_name,char *task_date){
    struct MaintenanceTask *new_task=(struct MaintenanceTask*)malloc(sizeof(struct MaintenanceTask));
    strcpy(new_task->task_name,task_name);
    strcpy(new_task->task_date,task_date);
    new_task->next=NULL;
    return new_task;
}
struct MaintenanceTask* insertTask(struct MaintenanceTask *head,char *task_name,char *task_date){
    struct MaintenanceTask* new_task = createTask(task_name, task_date);
    struct MaintenanceTask *temp=head;
    if(head==NULL){
        return new_task;
    }
    while(temp->next!=NULL){
        temp=temp->next;
    }
    temp->next=new_task;
    return head;
}
struct MaintenanceTask* deleteTask(struct MaintenaceTask *head){
    if (head == NULL) {
        printf("The schedule is empty. No task to delete.\n");
        return head;
    }
    struct MaintenanceTask *temp=head;
    head=head->next;
    free(temp);
    printf("Maintenance task completed and deleted from the schedule.\n");
```

```
    return head;

}
void displaySchedule(struct MaintenaceTask *head){
    if(head==NULL){
        printf("List empty");
    }
    struct MaintenanceTask* temp = head;
    while(temp!=NULL){
        printf("Task :%s ,Date :%s",temp->task_name,temp->task_date);
        temp=temp->next;
    }
}
int main() {
    struct MaintenanceTask* schedule = NULL;
    int choice;
    char task_name[100], task_date[20];

    while (1) {
        printf("\n--- Machine Maintenance Schedule ---\n");
        printf("1. Insert new maintenance task\n");
        printf("2. Delete completed maintenance task\n");
        printf("3. Display current maintenance schedule\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter the maintenance task name: ");
```

```
            getchar();
            fgets(task_name, sizeof(task_name), stdin);
            task_name[strcspn(task_name, "\n")] = '\0';
            printf("Enter the task date (YYYY-MM-DD): ");
            fgets(task_date, sizeof(task_date), stdin);
            task_date[strcspn(task_date, "\n")] = '\0';
            schedule = insertTask(schedule, task_name, task_date);
            printf("Maintenance task '%s' scheduled for %s.\n", task_name, task_date);
            break;

        case 2:
            schedule = deleteTask(schedule);
            break;

        case 3:
            displaySchedule(schedule);
            break;

        case 4:
            printf("Exiting the program.\n");
            return 0;

        default:
            printf("Invalid choice. Please try again.\n");
      }
   }

   return 0;
}
```

*4.*

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

struct Shift {
    char employee_name[100];
    char shift_time[20];
    struct Shift* next;
};

struct Shift* createShift(char* employee_name, char* shift_time) {
    struct Shift* new_shift = (struct Shift*)malloc(sizeof(struct Shift));
    strcpy(new_shift->employee_name, employee_name);
    strcpy(new_shift->shift_time, shift_time);
    new_shift->next = NULL;
    return new_shift;
}

struct Shift* insertShift(struct Shift* head, char* employee_name, char* shift_time) {
    struct Shift* new_shift = createShift(employee_name, shift_time);
    if (head == NULL) return new_shift;
    struct Shift* temp = head;
    while (temp->next != NULL) temp = temp->next;
    temp->next = new_shift;
    return head;
}
```

```
struct Shift* deleteShift(struct Shift* head) {
    if (head == NULL) return NULL;
    struct Shift* temp = head;
    head = head->next;
    free(temp);
    return head;
}

void displayShifts(struct Shift* head) {
    if (head == NULL) {
        printf("No shifts scheduled.\n");
        return;
    }
    struct Shift* temp = head;
    while (temp != NULL) {
        printf("Employee: %s, Shift Time: %s\n", temp->employee_name, temp->shift_time);
        temp = temp->next;
    }
}

int main() {
    struct Shift* schedule = NULL;
    schedule = insertShift(schedule, "John Doe", "9:00 AM - 5:00 PM");
    schedule = insertShift(schedule, "Jane Smith", "5:00 PM - 1:00 AM");

    displayShifts(schedule);

    schedule = deleteShift(schedule);
```

```
    displayShifts(schedule);
    return 0;
}
```

5.

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

struct Order {
    char order_id[50];
    char customer_name[100];
    struct Order* next;
};

struct Order* createOrder(char* order_id, char* customer_name) {
    struct Order* new_order = (struct Order*)malloc(sizeof(struct Order));
    strcpy(new_order->order_id, order_id);
    strcpy(new_order->customer_name, customer_name);
    new_order->next = NULL;
    return new_order;
}

struct Order* insertOrder(struct Order* head, char* order_id, char* customer_name) {
    struct Order* new_order = createOrder(order_id, customer_name);
    if (head == NULL) return new_order;
    struct Order* temp = head;
    while (temp->next != NULL) temp = temp->next;
    temp->next = new_order;
```

```
    return head;
}

struct Order* deleteOrder(struct Order* head) {
    if (head == NULL) return NULL;
    struct Order* temp = head;
    head = head->next;
    free(temp);
    return head;
}

void displayOrders(struct Order* head) {
    if (head == NULL) {
        printf("No orders in the system.\n");
        return;
    }
    struct Order* temp = head;
    while (temp != NULL) {
        printf("Order ID: %s, Customer: %s\n", temp->order_id, temp->customer_name);
        temp = temp->next;
    }
}

int main() {
    struct Order* orders = NULL;
    orders = insertOrder(orders, "ORD001", "Alice");
    orders = insertOrder(orders, "ORD002", "Bob");

    displayOrders(orders);
```

```
    orders = deleteOrder(orders);

    displayOrders(orders);
    return 0;
}

6.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Tool {
    char tool_name[100];
    char status[50];
    struct Tool* next;
};

struct Tool* createTool(char* tool_name, char* status) {
    struct Tool* new_tool = (struct Tool*)malloc(sizeof(struct Tool));
    strcpy(new_tool->tool_name, tool_name);
    strcpy(new_tool->status, status);
    new_tool->next = NULL;
    return new_tool;
}

struct Tool* insertTool(struct Tool* head, char* tool_name, char* status) {
    struct Tool* new_tool = createTool(tool_name, status);
    if (head == NULL) return new_tool;
```

```
    struct Tool* temp = head;
    while (temp->next != NULL) temp = temp->next;
    temp->next = new_tool;
    return head;
}

struct Tool* deleteTool(struct Tool* head) {
    if (head == NULL) return NULL;
    struct Tool* temp = head;
    head = head->next;
    free(temp);
    return head;
}

void displayTools(struct Tool* head) {
    if (head == NULL) {
        printf("No tools being tracked.\n");
        return;
    }
    struct Tool* temp = head;
    while (temp != NULL) {
        printf("Tool: %s, Status: %s\n", temp->tool_name, temp->status);
        temp = temp->next;
    }
}

int main() {
    struct Tool* tools = NULL;
    tools = insertTool(tools, "Wrench", "In use");
```

```
    tools = insertTool(tools, "Drill", "Available");

    displayTools(tools);

    tools = deleteTool(tools);

    displayTools(tools);
    return 0;
}

7.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct AssemblyStage {
    char stage_name[100];
    struct AssemblyStage* next;
};

struct AssemblyStage* createStage(char* stage_name) {
    struct AssemblyStage* new_stage = (struct AssemblyStage*)malloc(sizeof(struct AssemblyStage));
    strcpy(new_stage->stage_name, stage_name);
    new_stage->next = NULL;
    return new_stage;
}

struct AssemblyStage* insertStage(struct AssemblyStage* head, char* stage_name) {
    struct AssemblyStage* new_stage = createStage(stage_name);
```

```
    if (head == NULL) return new_stage;
    struct AssemblyStage* temp = head;
    while (temp->next != NULL) temp = temp->next;
    temp->next = new_stage;
    return head;
}

struct AssemblyStage* deleteStage(struct AssemblyStage* head) {
    if (head == NULL) return NULL;
    struct AssemblyStage* temp = head;
    head = head->next;
    free(temp);
    return head;
}

void displayStages(struct AssemblyStage* head) {
    if (head == NULL) {
        printf("No assembly stages.\n");
        return;
    }
    struct AssemblyStage* temp = head;
    while (temp != NULL) {
        printf("Stage: %s\n", temp->stage_name);
        temp = temp->next;
    }
}

int main() {
    struct AssemblyStage* stages = NULL;
```

```
    stages = insertStage(stages, "Design");
    stages = insertStage(stages, "Manufacturing");

    displayStages(stages);

    stages = deleteStage(stages);

    displayStages(stages);
    return 0;
}
```

8.

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

struct Order {
    char order_id[50];
    char customer_name[100];
    struct Order* next;
};

struct Order* createOrder(char* order_id, char* customer_name) {
    struct Order* new_order = (struct Order*)malloc(sizeof(struct Order));
    strcpy(new_order->order_id, order_id);
    strcpy(new_order->customer_name, customer_name);
    new_order->next = NULL;
    return new_order;
}
```

```
struct Order* insertOrder(struct Order* head, char* order_id, char* customer_name) {
    struct Order* new_order = createOrder(order_id, customer_name);
    if (head == NULL) return new_order;
    struct Order* temp = head;
    while (temp->next != NULL) temp = temp->next;
    temp->next = new_order;
    return head;
}

struct Order* deleteOrder(struct Order* head) {
    if (head == NULL) return NULL;
    struct Order* temp = head;
    head = head->next;
    free(temp);
    return head;
}

void displayOrders(struct Order* head) {
    if (head == NULL) {
        printf("No orders in the system.\n");
        return;
    }
    struct Order* temp = head;
    while (temp != NULL) {
        printf("Order ID: %s, Customer: %s\n", temp->order_id, temp->customer_name);
        temp = temp->next;
    }
}
```

```
int main() {
    struct Order* orders = NULL;
    orders = insertOrder(orders, "ORD001", "Alice");
    orders = insertOrder(orders, "ORD002", "Bob");

    displayOrders(orders);

    orders = deleteOrder(orders);

    displayOrders(orders);
    return 0;
}

9.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Tool {
    char tool_name[100];
    char status[50];
    struct Tool* next;
};

struct Tool* createTool(char* tool_name, char* status) {
    struct Tool* new_tool = (struct Tool*)malloc(sizeof(struct Tool));
    strcpy(new_tool->tool_name, tool_name);
    strcpy(new_tool->status, status);
```

```
    new_tool->next = NULL;
    return new_tool;
}

struct Tool* insertTool(struct Tool* head, char* tool_name, char* status) {
    struct Tool* new_tool = createTool(tool_name, status);
    if (head == NULL) return new_tool;
    struct Tool* temp = head;
    while (temp->next != NULL) temp = temp->next;
    temp->next = new_tool;
    return head;
}

struct Tool* deleteTool(struct Tool* head) {
    if (head == NULL) return NULL;
    struct Tool* temp = head;
    head = head->next;
    free(temp);
    return head;
}

void displayTools(struct Tool* head) {
    if (head == NULL) {
        printf("No tools being tracked.\n");
        return;
    }
    struct Tool* temp = head;
    while (temp != NULL) {
        printf("Tool: %s, Status: %s\n", temp->tool_name, temp->status);
```

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

int main() {
    struct Tool* tools = NULL;
    tools = insertTool(tools, "Wrench", "In use");
    tools = insertTool(tools, "Drill", "Available");

    displayTools(tools);

    tools = deleteTool(tools);

    displayTools(tools);
    return 0;
}

10.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Tool {
    char tool_name[100];
    char status[50];
    struct Tool* next;
};

struct Tool* createTool(char* tool_name, char* status) {
```

```
    struct Tool* new_tool = (struct Tool*)malloc(sizeof(struct Tool));
    strcpy(new_tool->tool_name, tool_name);
    strcpy(new_tool->status, status);
    new_tool->next = NULL;
    return new_tool;
}

struct Tool* insertTool(struct Tool* head, char* tool_name, char* status) {
    struct Tool* new_tool = createTool(tool_name, status);
    if (head == NULL) return new_tool;
    struct Tool* temp = head;
    while (temp->next != NULL) temp = temp->next;
    temp->next = new_tool;
    return head;
}

struct Tool* deleteTool(struct Tool* head) {
    if (head == NULL) return NULL;
    struct Tool* temp = head;
    head = head->next;
    free(temp);
    return head;
}

void displayTools(struct Tool* head) {
    if (head == NULL) {
        printf("No tools being tracked.\n");
        return;
    }
```

```
    struct Tool* temp = head;
    while (temp != NULL) {
        printf("Tool: %s, Status: %s\n", temp->tool_name, temp->status);
        temp = temp->next;
    }
}

int main() {
    struct Tool* tools = NULL;
    tools = insertTool(tools, "Wrench", "In use");
    tools = insertTool(tools, "Drill", "Available");

    displayTools(tools);

    tools = deleteTool(tools);

    displayTools(tools);
    return 0;
}
```

11.

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

struct AssemblyStage {
    char stage_name[100];
    struct AssemblyStage* next;
};
```

```
struct AssemblyStage* createStage(char* stage_name) {
    struct AssemblyStage* new_stage = (struct AssemblyStage*)malloc(sizeof(struct AssemblyStage));
    strcpy(new_stage->stage_name, stage_name);
    new_stage->next = NULL;
    return new_stage;
}

struct AssemblyStage* insertStage(struct AssemblyStage* head, char* stage_name) {
    struct AssemblyStage* new_stage = createStage(stage_name);
    if (head == NULL) return new_stage;
    struct AssemblyStage* temp = head;
    while (temp->next != NULL) temp = temp->next;
    temp->next = new_stage;
    return head;
}

struct AssemblyStage* deleteStage(struct AssemblyStage* head) {
    if (head == NULL) return NULL;
    struct AssemblyStage* temp = head;
    head = head->next;
    free(temp);
    return head;
}

void displayStages(struct AssemblyStage* head) {
    if (head == NULL) {
        printf("No assembly stages.\n");
        return;
```

```
    }
    struct AssemblyStage* temp = head;
    while (temp != NULL) {
        printf("Stage: %s\n", temp->stage_name);
        temp = temp->next;
    }
}

int main() {
    struct AssemblyStage* stages = NULL;
    stages = insertStage(stages, "Design");
    stages = insertStage(stages, "Manufacturing");

    displayStages(stages);

    stages = deleteStage(stages);

    displayStages(stages);
    return 0;
}

12.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct ChecklistItem {
    char item_name[100];
    struct ChecklistItem* next;
```

```
};

struct ChecklistItem* createItem(char* item_name) {
    struct ChecklistItem* new_item = (struct ChecklistItem*)malloc(sizeof(struct ChecklistItem));
    strcpy(new_item->item_name, item_name);
    new_item->next = NULL;
    return new_item;
}

struct ChecklistItem* insertItem(struct ChecklistItem* head, char* item_name) {
    struct ChecklistItem* new_item = createItem(item_name);
    if (head == NULL) return new_item;
    struct ChecklistItem* temp = head;
    while (temp->next != NULL) temp = temp->next;
    temp->next = new_item;
    return head;
}

struct ChecklistItem* deleteItem(struct ChecklistItem* head) {
    if (head == NULL) return NULL;
    struct ChecklistItem* temp = head;
    head = head->next;
    free(temp);
    return head;
}

void displayItems(struct ChecklistItem* head) {
    if (head == NULL) {
        printf("No checklist items.\n");
```

```
        return;
    }
    struct ChecklistItem* temp = head;
    while (temp != NULL) {
        printf("Checklist Item: %s\n", temp->item_name);
        temp = temp->next;
    }
}

int main() {
    struct ChecklistItem* checklist = NULL;
    checklist = insertItem(checklist, "Check Product Dimensions");
    checklist = insertItem(checklist, "Test Functionality");

    displayItems(checklist);

    checklist = deleteItem(checklist);

    displayItems(checklist);
    return 0;
}
```

13.

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

struct PackagingTask {
    char task_name[100];
```

```
    struct PackagingTask* next;
};

struct PackagingTask* createPackagingTask(char* task_name) {
    struct PackagingTask* new_task = (struct PackagingTask*)malloc(sizeof(struct PackagingTask));
    strcpy(new_task->task_name, task_name);
    new_task->next = NULL;
    return new_task;
}

struct PackagingTask* insertPackagingTask(struct PackagingTask* head, char* task_name) {
    struct PackagingTask* new_task = createPackagingTask(task_name);
    if (head == NULL) return new_task;
    struct PackagingTask* temp = head;
    while (temp->next != NULL) temp = temp->next;
    temp->next = new_task;
    return head;
}

struct PackagingTask* deletePackagingTask(struct PackagingTask* head) {
    if (head == NULL) return NULL;
    struct PackagingTask* temp = head;
    head = head->next;
    free(temp);
    return head;
}

void displayPackagingTasks(struct PackagingTask* head) {
    if (head == NULL) {
```

```
        printf("No tasks in the packaging schedule.\n");
        return;
    }
    struct PackagingTask* temp = head;
    while (temp != NULL) {
        printf("Packaging Task: %s\n", temp->task_name);
        temp = temp->next;
    }
}

int main() {
    struct PackagingTask* tasks = NULL;
    tasks = insertPackagingTask(tasks, "Wrap Products");
    tasks = insertPackagingTask(tasks, "Label Products");

    displayPackagingTasks(tasks);

    tasks = deletePackagingTask(tasks);

    displayPackagingTasks(tasks);
    return 0;
}

14.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Defect {
```

```
    char defect_description[100];
    struct Defect* next;
};

struct Defect* createDefect(char* defect_description) {
    struct Defect* new_defect = (struct Defect*)malloc(sizeof(struct Defect));
    strcpy(new_defect->defect_description, defect_description);
    new_defect->next = NULL;
    return new_defect;
}

struct Defect* insertDefect(struct Defect* head, char* defect_description) {
    struct Defect* new_defect = createDefect(defect_description);
    if (head == NULL) return new_defect;
    struct Defect* temp = head;
    while (temp->next != NULL) temp = temp->next;
    temp->next = new_defect;
    return head;
}

struct Defect* deleteDefect(struct Defect* head) {
    if (head == NULL) return NULL;
    struct Defect* temp = head;
    head = head->next;
    free(temp);
    return head;
}

void displayDefects(struct Defect* head) {
```

```
    if (head == NULL) {
        printf("No defects reported.\n");
        return;
    }
    struct Defect* temp = head;
    while (temp != NULL) {
        printf("Defect: %s\n", temp->defect_description);
        temp = temp->next;
    }
}

int main() {
    struct Defect* defects = NULL;
    defects = insertDefect(defects, "Scratched Surface");
    defects = insertDefect(defects, "Faulty Wiring");

    displayDefects(defects);

    defects = deleteDefect(defects);

    displayDefects(defects);
    return 0;
}
```

15.

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

```
struct DispatchEntry {
    char product_name[100];
    char dispatch_date[20];
    struct DispatchEntry* next;
};

struct DispatchEntry* createDispatchEntry(char* product_name, char* dispatch_date) {
    struct DispatchEntry* new_entry = (struct DispatchEntry*)malloc(sizeof(struct DispatchEntry));
    strcpy(new_entry->product_name, product_name);
    strcpy(new_entry->dispatch_date, dispatch_date);
    new_entry->next = NULL;
    return new_entry;
}

struct DispatchEntry* insertDispatchEntry(struct DispatchEntry* head, char* product_name, char*
dispatch_date) {
    struct DispatchEntry* new_entry = createDispatchEntry(product_name, dispatch_date);
    if (head == NULL) return new_entry;
    struct DispatchEntry* temp = head;
    while (temp->next != NULL) temp = temp->next;
    temp->next = new_entry;
    return head;
}

struct DispatchEntry* deleteDispatchEntry(struct DispatchEntry* head) {
    if (head == NULL) return NULL;
    struct DispatchEntry* temp = head;
    head = head->next;
    free(temp);
```

```
    return head;
}

void displayDispatchEntries(struct DispatchEntry* head) {
    if (head == NULL) {
        printf("No dispatches in the system.\n");
        return;
    }
    struct DispatchEntry* temp = head;
    while (temp != NULL) {
        printf("Product: %s, Dispatch Date: %s\n", temp->product_name, temp->dispatch_date);
        temp = temp->next;
    }
}

int main() {
    struct DispatchEntry* dispatches = NULL;
    dispatches = insertDispatchEntry(dispatches, "Product A", "2025-01-01");
    dispatches = insertDispatchEntry(dispatches, "Product B", "2025-01-02");

    displayDispatchEntries(dispatches);

    dispatches = deleteDispatchEntry(dispatches);

    displayDispatchEntries(dispatches);
    return 0;
}

*/
```

---

```
/*
1.Team roaster management
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<stdbool.h>
struct player{
   char name[30];
   int number;
   struct player *next;
};
struct player* createPlayer(char *name,int number){
   struct player *new_player=(struct player*)malloc(sizeof(struct player));
   strcpy(new_player->name,name);
   new_player->number=number;
   new_player->next=NULL;
   return new_player;
}
struct player* insertPlayer(struct player *head,char *name,int number){
   struct player *new_player=createPlayer(name,number);
   if(head==NULL){
      return new_player;
   }
   struct player *temp=head;
   while(temp->next!=NULL){
      temp=temp->next;
   }
```

```
    temp->next=new_player;
    return head;

}
struct player* deletePlayer(struct player  *head,char *name){
    if(head==NULL){
        printf("Empty List!");
    }
    struct player* temp=head;
    struct player* prev=NULL;
    if(strcmp(head->name,name)==0){
        head=head->next;
        free(temp);
        printf("Player '%s' removed from the List.\n", name);
        return head;
    }
    while(temp!=NULL && strcmp(head->name,name)!=0){
        temp=temp->next;
        prev=temp;
    }
    if(temp==NULL){
        printf("Player not found !");
    }
    prev->next=temp->next;
    free(temp);
    printf("Player '%s' removed from the List.\n", name);
}
void displayRoster(struct player  *head){
    struct player *temp=head;
```

```
    while(temp!=NULL){
        printf("player->%s, Number->%d",temp->name,temp->number);
        temp=temp->next;
    }
}
int main() {
    struct player* roster = NULL;
    int choice;
    char name[100];
    int number;

    while (1) {
        printf("\n--- Team Roster Management ---\n");
        printf("1. Add a new player to the team\n");
        printf("2. Remove a player who leaves the team\n");
        printf("3. Display the current team roster\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter player name: ");
                getchar();
                fgets(name, sizeof(name), stdin);
                name[strcspn(name, "\n")] = '\0';
                printf("Enter player jersey number: ");
                scanf("%d", &number);
                roster = insertPlayer(roster, name, number);
                printf("Player '%s' added to the roster.\n", name);
```

```
            break;
        case 2:
            printf("Enter player name to remove: ");
            getchar();
            fgets(name, sizeof(name), stdin);
            name[strcspn(name, "\n")] = '\0';
            roster = deletePlayer(roster, name);
            break;
        case 3:
            displayRoster(roster);
            break;
        case 4:
            printf("Exiting the program.\n");
            return 0;
        default:
            printf("Invalid choice. Please try again.\n");
      }
   }

   return 0;
}
```