

Task1

•The code:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_TASKS 100
typedef struct
{
    int id;
    char description[100];
} Task;
Task taskList[MAX_TASKS];
int numTasks = 0;
int isIdExist(int id) {
    for (int i = 0; i < numTasks; i++) {
        if (taskList[i].id == id) {
            return 1;
        }
    }
    return 0;
}
void add_Task() {
    if (numTasks >= MAX_TASKS) {
        printf("Task list is full. Cannot add more tasks.\n");
        return;
    }
```

```

Task newTask;
printf("Enter task description: ");
scanf(" %[\n]", newTask.description);
newTask.id = numTasks + 1;
taskList[numTasks++] = newTask;
printf("Task added successfully!\n\n");
}

int view_Tasks() {
    if (numTasks == 0) {
        printf("No tasks found.\n");
        return;
    }
    printf("Current Tasks:\n");
    for (int i = 0; i < numTasks; i++) {
        printf("Task ID: %d\nDescription: %s\n\n", taskList[i].id, taskList[i].description);
    }
}

void remove_Task() {
    if (numTasks == 0) {
        printf("No tasks found.\n");
        return;
    }
    int taskId;
    printf("Enter the task ID to remove: ");
    scanf("%d", &taskId);
    int foundIndex = -1;
    for (int i = 0; i < numTasks; i++) {
        if (taskList[i].id == taskId) {

```

```

foundIndex = i;
break;
}
}
if (foundIndex == -1) {
printf("Task with ID %d not found.\n", taskId);
return;
}
for (int i = foundIndex; i < numTasks - 1; i++) {
taskList[i] = taskList[i + 1];
}
numTasks--;
printf("Task removed successfully!\n\n");
}
int main() {

printf("Minions Task Manager\n");
printf("1. Add Task\n");
printf("2. View Tasks\n");
printf("3. Remove Task\n");
printf("4. Exit\n\n");

while(1){

int option;
printf("Select an option: ");
scanf("%d",&option);

```

```
switch (option) {  
    case 1:  
        add_Task();  
        break;  
    case 2:  
        view_Tasks();  
        break;  
    case 3:  
        remove_Task();  
        break;  
    case 4:  
        printf("Exiting Minions Task Manager. Have a great day!\n\n");  
        exit(0);  
    default:  
        printf("Invalid choice. Please try again.\n");  
}  
}  
return 0;  
}
```

•Screen of the output:

C:\Users\LENOVO\Documents\refdre.exe

```
Minions Task Manager
1. Add Task
2. View Tasks
3. Remove Task
4. Exit

Select an option: 1
Enter task description: Prepare moon launch materials
Task added successfully!

Select an option: 2
Current Tasks:
Task ID: 1
Description: Prepare moon launch materials

Select an option: 1
Enter task description: Check spaceship fuel levels
Task added successfully!

Select an option: 2
Current Tasks:
Task ID: 1
Description: Prepare moon launch materials

Task ID: 2
Description: Check spaceship fuel levels

Select an option: 3
Enter the task ID to remove: 1
Task removed successfully!

Select an option: 2
Current Tasks:
Task ID: 2
Description: Check spaceship fuel levels

Select an option: 4
Exiting Minions Task Manager. Have a great day!

Process returned 0 (0x0)   execution time : 92.430 s
Press any key to continue.
```

Bonus

•The code:

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

```
#include <stdbool.h>
```

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

```
#define MAX_TASKS 100
```

```
typedef struct
```

```
{
```

```
    int id;
```

```
    char description[100];
```

```
    bool completed;
```

```
} Task;
```

```
Task taskList[MAX_TASKS];
```

```
int numTasks = 0;
```

```
void add_Task()
```

```
{
```

```
    if (numTasks == MAX_TASKS)
```

```
    {
```

```
        printf("Task list is full. Cannot add more tasks.\n");
```

```
        return;
```

```
}
```

```
Task newTask;
```

```
printf("Enter task description: ");
```

```
fgets(newTask.description,100, stdin);
```

```
newTask.description[strcspn(newTask.description, "\n")] = '\0'; // Remove trailing  
newline
```

```
newTask.completed = false;
```

```
newTask.id = numTasks + 1;
```

```
taskList[numTasks] = newTask;
```

```
numTasks++;
```

```
printf("Task added successfully.\n\n");
```

```
}
```

```
void view_Tasks() {
```

```
    if (numTasks == 0) {
```

```
        printf("No tasks found.\n");
```

```
        return;
```

```
    }
```

```
    printf("ID\tDescription\t\tCompleted\n");
```

```
    printf("-----\n");
```

```
    for (int i = 0; i < numTasks; i++) {
```

```
        printf("%d\t%s\t\t%s\n", taskList[i].id, taskList[i].description, taskList[i].completed ?  
"Yes" : "No");
```

```
    }
```

```
}
```

```

void remove_Task() {
    if (numTasks == 0) {
        printf("No tasks found.\n");
        return;
    }

    int taskId;
    printf("Enter the ID of the task to remove: ");
    scanf("%d", &taskId);

    int taskIndex = -1;
    for (int i = 0; i < numTasks; i++) {
        if (taskList[i].id == taskId) {
            taskIndex = i;
            break;
        }
    }

    if (taskIndex == -1) {
        printf("Task with ID %d not found.\n", taskId);
        return;
    }

    // Shift remaining tasks to fill the gap
    for (int i = taskIndex; i < numTasks - 1; i++) {
        taskList[i] = taskList[i + 1];
    }
}

```



```
    numTasks--;  
    printf("Task with ID %d removed successfully.\n\n", taskId);  
}
```

```
void mark_Task_As_Completed() {  
    if (numTasks == 0) {  
        printf("No tasks found.\n\n");  
        return;  
    }  
}
```

```
int taskId;  
printf("Enter the ID of the task to mark as completed: ");  
scanf("%d", &taskId);
```

```
for (int i = 0; i < numTasks; i++) {  
    if (taskList[i].id == taskId) {  
        taskList[i].completed = true;  
        printf("Task with ID %d marked as completed.\n\n", taskId);  
        return;  
    }  
}
```

```
printf("Task with ID %d not found.\n", taskId);  
}
```

```
void view_Completed_Tasks() {  
    printf("Completed Tasks:\n\n");
```

```
printf("ID\tDescription\n\n");  
printf("-----\n");
```

```
bool foundCompletedTasks = false;  
for (int i = 0; i < numTasks; i++) {  
    if (taskList[i].completed) {  
        printf("%d\t%s\n", taskList[i].id, taskList[i].description);  
        foundCompletedTasks = true;  
    }  
}
```

```
if (!foundCompletedTasks) {  
    printf("No completed tasks found.\n\n");  
}  
}
```

```
void view_Incomplete_Tasks() {  
    printf("Incomplete Tasks:\n");  
    printf("ID\tDescription\n");  
    printf("-----\n");
```

```
bool foundIncompleteTasks = false;  
for (int i = 0; i < numTasks; i++) {  
    if (!taskList[i].completed) {  
        printf("%d\t%s\n", taskList[i].id, taskList[i].description);  
        foundIncompleteTasks = true;  
    }  
}
```

```
    if (!foundIncompleteTasks) {  
        printf("No incomplete tasks found.\n\n");  
    }  
}
```

```
int main() {  
    printf("\nTask Management System\n");  
    printf("1. Add Task\n");  
    printf("2. View Tasks\n");  
    printf("3. Remove Task\n");  
    printf("4. Mark Task as Completed\n");  
    printf("5. View Completed Tasks\n");  
    printf("6. View Incomplete Tasks\n");  
    printf("7. Exit\n\n\n");
```

```
    int option;
```

```
    while (1)  
    {  
        printf("Select an option: ");  
        scanf("%d", &option);  
        getchar();
```

```
        switch (option) {  
            case 1:  
                add_Task();  
                break;
```

```
case 2:
    view_Tasks();
    break;
case 3:
    remove_Task();
    break;
case 4:
    mark_Task_As_Completed();
    break;
case 5:
    view_Completed_Tasks();
    break;
case 6:
    view_Incomplete_Tasks();
    break;
case 7:
    exit(0);
default:
    printf("Invalid choice. Please try again.\n");
}
}

return 0;
}
```

•Screen of the output:

C:\Users\LENOVO\Documents\34131.exe

```
Task Management System
1. Add Task
2. View Tasks
3. Remove Task
4. Mark Task as Completed
5. View Completed Tasks
6. View Incomplete Tasks
7. Exit

Select an option: 1
Enter task description: Prepare moon launch materials
Task added successfully.

Select an option: 1
Enter task description: Check spaceship fuel levels
Task added successfully.

Select an option: 2
ID      Description      Completed
-----
1       Prepare moon launch materials    No
2       Check spaceship fuel levels     No
Select an option: 3
Enter the ID of the task to remove: 2
Task with ID 2 removed successfully.

Select an option: 2
ID      Description      Completed
-----
1       Prepare moon launch materials    No
Select an option: 5
Completed Tasks:

ID      Description
-----
No completed tasks found.

Select an option: 6
Incomplete Tasks:
ID      Description
-----
1       Prepare moon launch materials
Select an option: 7

Process returned 0 (0x0)   execution time : 84.358 s
Press any key to continue.
```