Requirements

1. Define Data Types

1. **Printer Job Structure:**
   Create a structure PrintJob to represent a single 3D printing task.
   Fields:
   - jobID (integer): Unique ID for the print job.
   - fileName (string): Name of the file to be printed (e.g., "model.stl").
   - materialType (string): Material used for the job (e.g., PLA, ABS).
   - printTime (float): Estimated print time in hours.
   - status (string): Current status of the job (e.g., "In Progress", "Completed").
2. **Union for Printer Mode:**
   Define a union PrinterMode to represent the 3D printer's operational mode.
   Fields:
   - temperature (float): Current temperature of the print head (in °C).
   - calibrationStatus (string): Status of printer calibration (e.g., "Calibrated", "Not Calibrated").
3. **Printer Configuration Structure:**
   Define a structure PrinterConfig to store the 3D printer's configurations.
   Fields:
   - printerID (integer): Unique ID for the printer.
   - maxTemperature (float): Maximum allowable print head temperature (in °C).
   - bedSize (float): Size of the printer bed (in cm²).
   - mode (PrinterMode union): Current mode details of the printer.
4. **Typedef for Simplification:**
   Use typedef to create aliases for PrintJob, PrinterMode, and PrinterConfig for better code readability.

2. Features to Implement

1. **Dynamic Memory Allocation:**
   - Allocate memory dynamically for an array of PrintJob structures to handle a variable number of print jobs.
   - Allocate memory dynamically for multiple PrinterConfig structures to manage multiple printers.
2. **Input and Output:**
   - Input the details of each print job, including its material, estimated time, and current status.
   - Input the configuration of printers, such as bed size and max temperature.
   - Display all print jobs and printer configurations.
3. **Monitoring and Analysis:**
   - Monitor and display the printer's current operational mode, either temperature or calibration status.
   - Identify jobs exceeding a certain print time threshold and display their details.
4. **Sorting and Searching:**
   - Sort print jobs by their estimated print time in descending order.

- Search for a print job by its jobID and display its details.

5. **Job Management:**
   - Allow updating the status of a print job (e.g., from "In Progress" to "Completed").
   - Remove completed print jobs dynamically and reallocate memory for the remaining jobs.

Example Program Flow

1. **Menu-Driven Interface:**
   Provide a menu with the following options:
   - Add Print Job Details
   - View All Print Jobs
   - Update Job Status
   - Remove Completed Jobs
   - Sort Print Jobs by Estimated Time
   - Display Printer Configurations
   - Exit
   -

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

typedef struct {
    int jobid;
    char filename[50];
    char materialtype[30];
    float printtime;
    char status[20];
} Printjob;

typedef union {
    float temperature;
    char calibrationstatus[20];
} PrinterMode;

typedef struct {
    int printerid;
    float maxtemperature;
    float bedsize;
    PrinterMode mode;
    char modetype[10];
```

```
} Printerconfig;

Printjob *jobs = NULL;
int jobcount = 0;

void addprintjob();
void viewallprintjobs();
void updatejobstatus();
void removecompletedjobs();
void sortprintjobs();
void menu();

int main() {
    int choice;
    do {
        menu();
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                addprintjob();
                break;
            case 2:
                viewallprintjobs();
                break;
            case 3:
                updatejobstatus();
                break;
            case 4:
                removecompletedjobs();
                break;
            case 5:
                sortprintjobs();
                break;
            case 6:
                printf("Exiting program.\n");
                break;
            default:
                printf("Invalid choice. Try again.\n");
        }
    } while (choice != 6);

    free(jobs);
    return 0;
}


void menu() {
```

```
    printf("\n----- Menu -----\n");
    printf("1. Add Print Job\n");
    printf("2. View All Print Jobs\n");
    printf("3. Update Job Status\n");
    printf("4. Remove Completed Jobs\n");
    printf("5. Sort Print Jobs by Estimated Time\n");
    printf("6. Exit\n");
}

void addprintjob() {
    printf("\nEnter the number of print jobs to add: ");
    int n;
    scanf("%d", &n);

    jobs = realloc(jobs, (jobcount + n) * sizeof(Printjob));
    if (!jobs) {
        printf("Memory allocation failed.\n");
        return;
    }

    for (int i = 0; i < n; i++) {
    printf("\n--- Print Job %d ---\n", jobcount + i + 1);
    printf("Enter Job ID: \n");
    scanf("%d", &jobs[jobcount + i].jobid);
    printf("Enter File Name: \n");
    scanf("%s", jobs[jobcount + i].filename);
    printf("Enter Material Type: \n");
    scanf("%s", jobs[jobcount + i].materialtype);
    printf("Enter Print Time (hours): \n");
    scanf("%f", &jobs[jobcount + i].printtime);
    printf("Enter Status: ");
    scanf("%s", jobs[jobcount + i].status);
}

jobcount += n;

}

void viewallprintjobs() {
    printf("\n--- All Print Jobs ---\n");
    if (jobcount == 0) {
        printf("No print jobs available.\n");
        return;
    }
    for (int i = 0; i < jobcount; i++) {
        printf("\nJob ID: %d\n", jobs[i].jobid);
        printf("File Name: %s\n", jobs[i].filename);
        printf("Material Type: %s\n", jobs[i].materialtype);
```

```
        printf("Print Time: %.2f hours\n", jobs[i].printtime);
        printf("Status: %s\n", jobs[i].status);
    }
}

void updatejobstatus() {
    printf("\nEnter Job ID to update status: ");
    int id, found = 0;
    scanf("%d", &id);
    for (int i = 0; i < jobcount; i++) {
        if (jobs[i].jobid == id) {
            printf("Enter new status: ");
            scanf("%s", jobs[i].status);
            printf("Status updated successfully.\n");
            found = 1;
            break;
        }
    }
    if (!found) {
        printf("Job ID not found.\n");
    }
}

void removecompletedjobs() {
    int newCount = 0;
    for (int i = 0; i < jobcount; i++) {
        if (strcmp(jobs[i].status, "Completed") != 0) {
            jobs[newCount++] = jobs[i];
        }
    }
    jobcount = newCount;
    jobs = realloc(jobs, jobcount * sizeof(Printjob));
    printf("Completed jobs removed.\n");

}

void sortprintjobs() {
    for (int i = 0; i < jobcount - 1; i++) {
        for (int j = i + 1; j < jobcount; j++) {
            if (jobs[i].printtime < jobs[j].printtime) {
                Printjob temp = jobs[i];
                jobs[i] = jobs[j];
                jobs[j] = temp;
            }
        }
    }
    printf("Print jobs sorted by estimated time in descending order.\n");
}
```

o/p

```
----- Menu -----
1. Add Print Job
2. View All Print Jobs
3. Update Job Status
4. Remove Completed Jobs
5. Sort Print Jobs by Estimated Time
6. Exit

Enter your choice: 1

Enter the number of print jobs to add: 2

--- Print Job 1 ---
Enter Job ID:
1
Enter File Name:
abc
Enter Material Type:
ABS
Enter Print Time (hours):
2
Enter Status: inprogress

--- Print Job 2 ---
Enter Job ID:
2
Enter File Name:
xyz
Enter Material Type:
PLA
Enter Print Time (hours):
4.5
Enter Status: completed

----- Menu -----
1. Add Print Job
2. View All Print Jobs
```

```
----- Menu -----
1. Add Print Job
2. View All Print Jobs
3. Update Job Status
4. Remove Completed Jobs
5. Sort Print Jobs by Estimated Time
6. Exit

Enter your choice: 2

--- All Print Jobs ---

Job ID: 1
File Name: abc
Material Type: ABS
Print Time: 2.00 hours
Status: inprogress

Job ID: 2
File Name: xyz
Material Type: PLA
Print Time: 4.50 hours
Status: completed

----- Menu -----
1. Add Print Job
2. View All Print Jobs
3. Update Job Status
4. Remove Completed Jobs
5. Sort Print Jobs by Estimated Time
6. Exit

Enter your choice: 3

Enter Job ID to update status: 1
Enter new status: completed
Status updated successfully.
```

```
Status updated successfully.

----- Menu -----
1. Add Print Job
2. View All Print Jobs
3. Update Job Status
4. Remove Completed Jobs
5. Sort Print Jobs by Estimated Time
6. Exit

Enter your choice: 4
Completed jobs removed.

----- Menu -----
1. Add Print Job
2. View All Print Jobs
3. Update Job Status
4. Remove Completed Jobs
5. Sort Print Jobs by Estimated Time
6. Exit

Enter your choice: 5
Print jobs sorted by estimated time in descending order.

----- Menu -----
1. Add Print Job
2. View All Print Jobs
3. Update Job Status
4. Remove Completed Jobs
5. Sort Print Jobs by Estimated Time
6. Exit

Enter your choice: 6
Exiting program.
```