----------Structures and unions----------

1.union for mixed data

```
#include<stdio.h>
union{
   int a;
   char b;
}data;
int main(){
   union data;
   data.a=10;
   data.b='a';
   printf("the value of d.a is %d\n",data.a);
   printf("the value of d.b is %c",data.b);
   return 0;
}
```

2.student details

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

union {
   int roll_no;
   char name[30];
} data;

int main(){
   printf("Enter the name of student: ");
   getchar();
   fgets(data.name, 30, stdin);
```

```
    data.name[strcspn(data.name, "\n")] = '\0';

    printf("Enter the roll number: ");
    scanf("%d", &data.roll_no);

    printf("The student name is: %s\n", data.name);
    printf("Student roll number is: %d\n", data.roll_no);

    return 0;
}
```

3.Union for Measurement Units

```
#include<stdio.h>
union{
    float distance;
}data1;
int main(){
    printf("Enter the distance in km");
    scanf("%f",&data1.distance);
    printf("the data in miles is %.2f",data1.distance * 0.62);
}
```

4. union for shape dimensions

```
#include<stdio.h>
#include<stdbool.h>
union{
    float radius;
    float length;
    float width;
```

```
}data;
int main(){
   int user_choice;
   printf("1.Circle\n2.Rectangle\n3.Square\nEnter your choice :");
   scanf("%d",&user_choice);
   switch(user_choice){
      case 1:
         printf("Enter the radius of circle :");
         scanf("%f",&data.radius);
         printf("The area of circle is %.2f",3.14 * (data.radius * data.radius));
         break;
      case 2:
         printf("Enter the length :");
         scanf("%f",&data.length);
         printf("enter the width :");
         scanf("%f",&data.width);
         printf("the area is %.2f",data.length * data.width);
         break;
      case 3:
         printf("enter the side of square :");
         scanf("%f",&data.length);
         printf("the area of square is %.2f",data.length * data.length);
         break;
   }
   return 0;
}
```

5.union for employee data

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

```
union Employee {
    int employeeID;
    float salary;
};

int main() {
    union Employee emp;
    int choice;

    printf("Enter 1 to input Employee ID or 2 to input Salary: ");
    scanf("%d", &choice);

    if(choice == 1) {
        printf("Enter Employee ID: ");
        scanf("%d", &emp.employeeID);
        printf("Employee ID: %d\n", emp.employeeID);
    } else if(choice == 2) {
        printf("Enter Employee Salary: ");
        scanf("%f", &emp.salary);
        printf("Employee Salary: %.2f\n", emp.salary);
    } else {
        printf("Invalid choice.\n");
    }

    return 0;
}
```

6.Union for Sensor Data

```c
#include<stdio.h>

union SensorData {
   float temperature;
   float pressure;
};

int main() {
   union SensorData sensor;
   int choice;

   printf("Enter 1 to input temperature data or 2 to input pressure data: ");
   scanf("%d", &choice);

   if(choice == 1) {
      printf("Enter temperature reading: ");
      scanf("%f", &sensor.temperature);
      printf("Temperature: %.2f°C\n", sensor.temperature);
   } else if(choice == 2) {
      printf("Enter pressure reading: ");
      scanf("%f", &sensor.pressure);
      printf("Pressure: %.2f Pa\n", sensor.pressure);
   } else {
      printf("Invalid choice.\n");
   }

   return 0;
}
```

7.Union for Bank Account Information

```
#include<stdio.h>

union BankAccount {
   int accountNumber;
   float balance;
};

int main() {
   union BankAccount account;
   int choice;

   printf("Enter 1 to input Account Number or 2 to input Balance: ");
   scanf("%d", &choice);

   if(choice == 1) {
      printf("Enter Account Number: ");
      scanf("%d", &account.accountNumber);
      printf("Account Number: %d\n", account.accountNumber);
   } else if(choice == 2) {
      printf("Enter Balance: ");
      scanf("%f", &account.balance);
      printf("Account Balance: %.2f\n", account.balance);
   } else {
      printf("Invalid choice.\n");
   }

   return 0;
}
```

8.Union for Vehicle Information

```
#include<stdio.h>
#include<stdbool.h>
union details{
  int reg_no;
  float fuel_capacity;
};

int main(){
    union details det;
    int user_input;
    printf("1.Vechile registration number\n2.fuel Capacity\nInput either one of the data :");
    scanf("%d",&user_input);
    switch(user_input){
        case 1:
            printf("Enter the Vechile registration number :");
            scanf("%d",&det.reg_no);
            printf("the registration number is %d",det.reg_no);
            break;
        case 2:
            printf("Enter the Vechile registration number :");
            scanf("%f",&det.fuel_capacity);
            printf("the registration number is %f",det.fuel_capacity);
            break;
    }
    return 0;
}
```

9.union for exam results

```
#include<stdio.h>

union ExamResults {
   int marks;
   char grade;
};

int main() {
   union ExamResults result;
   int user_input;

   printf("1. Enter Marks\n2. Enter Grade\nInput either one of the data: ");
   scanf("%d", &user_input);

   switch(user_input) {
      case 1:
         printf("Enter the Marks: ");
         scanf("%d", &result.marks);
         printf("The Marks are: %d\n", result.marks);
         break;
      case 2:
         printf("Enter the Grade: ");
         getchar();
         scanf("%c", &result.grade);
         printf("The Grade is: %c\n", result.grade);
         break;
      default:
         printf("Invalid choice!\n");
```

```
        break;
    }

    return 0;
}
```

10. Union for Currency Conversion

```
#include<stdio.h>

union Currency {
    float USD;
    float EUR;
};

int main() {
    union Currency currency;
    int choice;
    float conversionRate = 0.85;

    printf("1. Enter USD\n2. Enter EUR\nInput either one of the data: ");
    scanf("%d", &choice);

    switch(choice) {
        case 1:
            printf("Enter the amount in USD: ");
            scanf("%f", &currency.USD);
            printf("The equivalent amount in EUR: %.2f\n", currency.USD * conversionRate);
            break;
        case 2:
```

```
        printf("Enter the amount in EUR: ");
        scanf("%f", &currency.EUR);
        printf("The equivalent amount in USD: %.2f\n", currency.EUR / conversionRate);
        break;
    default:
        printf("Invalid choice!\n");
        break;
  }

  return 0;
```

----------Tasks----------

```
1.Aircraft fleet management
#include <stdio.h>
#include <string.h>
#include <stdbool.h>

typedef struct airplanes {
  int id;
  char model[30];
  int capacity;
  char status;
} planes;

void Enter_flight_details(planes *arr, int *n) {
  printf("\nEnter the number of flights to input: ");
  scanf("%d", n);
  for (int i = 0; i < *n; i++) {
    printf("Enter the data for flight %d\n", i + 1);
    printf("Enter the flight id: ");
```

```
        scanf("%d", &arr[i].id);
        printf("Enter the flight model: ");
        getchar();  // Consume the newline character left by scanf
        fgets(arr[i].model, 30, stdin);
        arr[i].model[strcspn(arr[i].model, "\n")] = '\0';  // Remove newline
        printf("Enter the capacity: ");
        scanf("%d", &arr[i].capacity);
        printf("Enter the status: ");
        getchar();  // Consume the newline character left by scanf
        scanf("%c", &arr[i].status);
    }
    printf("Successfully entered flight details!\n");
}

void update_status(planes *arr, int *n, int *id1) {
    int found = 0;
    for (int i = 0; i < *n; i++) {
        if (arr[i].id == *id1) {
            char st;
            printf("Enter the new status: ");
            getchar();  // Consume the newline character
            scanf("%c", &st);
            arr[i].status = st;
            printf("Status updated for flight ID %d\n", arr[i].id);
            found = 1;
            break;
        }
    }
    if (!found) {
```

```
        printf("No match found for flight ID %d\n", *id1);
    }
}

void print_details(planes *arr, int *n, int *id2) {
    int found = 0;
    for (int i = 0; i < *n; i++) {
        if (arr[i].id == *id2) {
            printf("Aircraft ID: %d\n", arr[i].id);
            printf("Aircraft Model: %s\n", arr[i].model);
            printf("Aircraft Capacity: %d\n", arr[i].capacity);
            printf("Aircraft Status: %c\n", arr[i].status);
            found = 1;
            break;
        }
    }
    if (!found) {
        printf("No match found for flight ID %d\n", *id2);
    }
}

int main() {
    planes arr[100];
    int user_input, n = 0;
    bool is_on = true;

    while (is_on) {
        printf("1. Update status\n2. Display aircraft\n3. Enter flight details\nEnter your option: ");
        scanf("%d", &user_input);
```

```
    switch (user_input) {
        case 1:
            {
                int id1;
                printf("Enter the flight id to update status: ");
                scanf("%d", &id1);
                update_status(arr, &n, &id1);
            }
            break;

        case 2:
            {
                int id2;
                printf("Enter the flight id to display details: ");
                scanf("%d", &id2);
                print_details(arr, &n, &id2);
            }
            break;

        case 3:
            Enter_flight_details(arr, &n);
            break;

        default:
            printf("Invalid option! Please try again.\n");
    }
}
```

```
    return 0;
}
```

2.satellite data processing

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

#define TELEMETRY_LIMIT 100

typedef struct {
    float temperature;
    float velocity;
    float altitude;
} Telemetry;

typedef union {
    char imageData[256];
    Telemetry telemetryData;
} SatelliteData;

void processImageData(SatelliteData *data) {
    printf("Processing Image Data: %s\n", data->imageData);
}

void processTelemetryData(SatelliteData *data) {
    if (data->telemetryData.temperature > TELEMETRY_LIMIT) {
        printf("Warning: Temperature exceeds limit!\n");
    }
    if (data->telemetryData.velocity > TELEMETRY_LIMIT) {
```

```
        printf("Warning: Velocity exceeds limit!\n");
    }
    if (data->telemetryData.altitude > TELEMETRY_LIMIT) {
        printf("Warning: Altitude exceeds limit!\n");
    }
    printf("Telemetry Data - Temperature: %.2f, Velocity: %.2f, Altitude: %.2f\n",
           data->telemetryData.temperature,
           data->telemetryData.velocity,
           data->telemetryData.altitude);
}

int main() {
    SatelliteData data;
    int choice;

    while (1) {
        printf("\n1. Process Image Data\n2. Process Telemetry Data\n3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        if (choice == 1) {
            printf("Enter image data (up to 255 characters): ");
            getchar();
            fgets(data.imageData, sizeof(data.imageData), stdin);
            data.imageData[strcspn(data.imageData, "\n")] = '\0';
            processImageData(&data);
        } else if (choice == 2) {
            printf("Enter temperature: ");
            scanf("%f", &data.telemetryData.temperature);
```

```
            printf("Enter velocity: ");
            scanf("%f", &data.telemetryData.velocity);
            printf("Enter altitude: ");
            scanf("%f", &data.telemetryData.altitude);
            processTelemetryData(&data);
        } else if (choice == 3) {
            break;
        } else {
            printf("Invalid choice. Try again.\n");
        }
    }

    return 0;
}
```

3.Mission control system

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

#define MAX_MISSIONS 100

typedef struct {
    char name[30];
    int crewCount;
    int crewSize;
} CrewDetails;

typedef struct {
    char cargoType[30];
```

```
    float weight;
} CargoDetails;

typedef union {
    CrewDetails crew;
    CargoDetails cargo;
} Payload;

typedef struct {
    int missionID;
    char name[30];
    int duration;
    Payload payload;
    int missionType;
} Mission;

static int totalMissions = 0;

void addMission(Mission *missions) {
    printf("Enter mission ID: ");
    scanf("%d", &missions[totalMissions].missionID);

    printf("Enter mission name: ");
    getchar();
    fgets(missions[totalMissions].name, sizeof(missions[totalMissions].name), stdin);
    missions[totalMissions].name[strcspn(missions[totalMissions].name, "\n")] = '\0';

    printf("Enter mission duration (in days): ");
    scanf("%d", &missions[totalMissions].duration);
```

```
    printf("Select mission type (1 for Crew, 2 for Cargo): ");
    scanf("%d", &missions[totalMissions].missionType);

    if (missions[totalMissions].missionType == 1) {
        printf("Enter crew name: ");
        getchar();
        fgets(missions[totalMissions].payload.crew.name,
sizeof(missions[totalMissions].payload.crew.name), stdin);
        missions[totalMissions].payload.crew.name[strcspn(missions[totalMissions].payload.crew.name,
"\n")] = '\0';

        printf("Enter crew count: ");
        scanf("%d", &missions[totalMissions].payload.crew.crewCount);

        printf("Enter crew size: ");
        scanf("%d", &missions[totalMissions].payload.crew.crewSize);

    } else if (missions[totalMissions].missionType == 2) {
        printf("Enter cargo type: ");
        getchar();
        fgets(missions[totalMissions].payload.cargo.cargoType,
sizeof(missions[totalMissions].payload.cargo.cargoType), stdin);
        missions[totalMissions].payload.cargo.cargoType[strcspn(missions[totalMissions].payload.cargo.car
goType, "\n")] = '\0';

        printf("Enter cargo weight (in kg): ");
        scanf("%f", &missions[totalMissions].payload.cargo.weight);
    } else {
        printf("Invalid mission type\n");
```

```
        return;
    }

    totalMissions++;
}

void updateMissionDetails(Mission *missions) {
    int missionID;
    printf("Enter mission ID to update: ");
    scanf("%d", &missionID);

    for (int i = 0; i < totalMissions; i++) {
        if (missions[i].missionID == missionID) {
            printf("Enter new mission name: ");
            getchar();
            fgets(missions[i].name, sizeof(missions[i].name), stdin);
            missions[i].name[strcspn(missions[i].name, "\n")] = '\0';

            printf("Enter new mission duration (in days): ");
            scanf("%d", &missions[i].duration);

            printf("Select new mission type (1 for Crew, 2 for Cargo): ");
            scanf("%d", &missions[i].missionType);

            if (missions[i].missionType == 1) {
                printf("Enter new crew name: ");
                getchar();
                fgets(missions[i].payload.crew.name, sizeof(missions[i].payload.crew.name), stdin);
                missions[i].payload.crew.name[strcspn(missions[i].payload.crew.name, "\n")] = '\0';
```

```
            printf("Enter new crew count: ");
            scanf("%d", &missions[i].payload.crew.crewCount);

            printf("Enter new crew size: ");
            scanf("%d", &missions[i].payload.crew.crewSize);
        } else if (missions[i].missionType == 2) {
            printf("Enter new cargo type: ");
            getchar();
            fgets(missions[i].payload.cargo.cargoType, sizeof(missions[i].payload.cargo.cargoType), stdin);
            missions[i].payload.cargo.cargoType[strcspn(missions[i].payload.cargo.cargoType, "\n")] = '\0';

            printf("Enter new cargo weight (in kg): ");
            scanf("%f", &missions[i].payload.cargo.weight);
        } else {
            printf("Invalid mission type\n");
            return;
        }
        printf("Mission details updated successfully!\n");
        return;
    }
  }
  printf("Mission not found!\n");
}

void displayMissionSummaries(Mission *missions) {
  for (int i = 0; i < totalMissions; i++) {
    printf("\nMission ID: %d\n", missions[i].missionID);
    printf("Mission Name: %s\n", missions[i].name);
```

```
        printf("Mission Duration: %d days\n", missions[i].duration);

        if (missions[i].missionType == 1) {
            printf("Payload: Crew\n");
            printf("Crew Name: %s\n", missions[i].payload.crew.name);
            printf("Crew Count: %d\n", missions[i].payload.crew.crewCount);
            printf("Crew Size: %d\n", missions[i].payload.crew.crewSize);
        } else if (missions[i].missionType == 2) {
            printf("Payload: Cargo\n");
            printf("Cargo Type: %s\n", missions[i].payload.cargo.cargoType);
            printf("Cargo Weight: %.2f kg\n", missions[i].payload.cargo.weight);
        }
    }
}

int main() {
    Mission missions[MAX_MISSIONS];
    int choice;

    while (1) {
        printf("\nMission Control System\n");
        printf("1. Add Mission\n2. Update Mission Details\n3. Display Mission Summaries\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addMission(missions);
                break;
```

```
            case 2:
                updateMissionDetails(missions);
                break;
            case 3:
                displayMissionSummaries(missions);
                break;
            case 4:
                return 0;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }
}
```

4.Aircraft mainatance tracker

```
#include <stdio.h>
#include <string.h>
#include<stdbool.h>
#define MAX_LOGS 100
#define ROUTINE 1
#define EMERGENCY 2

typedef struct {
    int hours;
    int minutes;
} TimeDuration;

typedef struct {
    TimeDuration duration;
```

```
    char description[100];
} RoutineMaintenance;

typedef struct {
    char emergencyType[30];
    char severityLevel[20];
} EmergencyMaintenance;

typedef union {
    RoutineMaintenance routine;
    EmergencyMaintenance emergency;
} MaintenanceType;

typedef struct {
    int logID;
    int aircraftID;
    char date[20];
    MaintenanceType maintenance;
    int maintenanceType;  // 1 for routine, 2 for emergency
} MaintenanceLog;

const int ROUTINE_MAINTENANCE_FREQUENCY = 100;  // In flight hours
const int EMERGENCY_MAINTENANCE_FREQUENCY = 1;   // Emergency occurrences

static int totalLogs = 0;

void addMaintenanceLog(MaintenanceLog *logs) {
    printf("Enter maintenance log ID: ");
    scanf("%d", &logs[totalLogs].logID);
```

```
    printf("Enter aircraft ID: ");
    scanf("%d", &logs[totalLogs].aircraftID);

    printf("Enter maintenance date (DD/MM/YYYY): ");
    getchar();  // Consume newline character
    fgets(logs[totalLogs].date, sizeof(logs[totalLogs].date), stdin);
    logs[totalLogs].date[strcspn(logs[totalLogs].date, "\n")] = '\0';

    printf("Select maintenance type (1 for Routine, 2 for Emergency): ");
    scanf("%d", &logs[totalLogs].maintenanceType);

    if (logs[totalLogs].maintenanceType == ROUTINE) {
        printf("Enter maintenance duration (hours minutes): ");
        scanf("%d %d", &logs[totalLogs].maintenance.routine.duration.hours,
                &logs[totalLogs].maintenance.routine.duration.minutes);
        printf("Enter maintenance description: ");
        getchar();  // Consume newline character
        fgets(logs[totalLogs].maintenance.routine.description,
sizeof(logs[totalLogs].maintenance.routine.description), stdin);
        logs[totalLogs].maintenance.routine.description[strcspn(logs[totalLogs].maintenance.routine.descrip
tion, "\n")] = '\0';
    } else if (logs[totalLogs].maintenanceType == EMERGENCY) {
        printf("Enter emergency type: ");
        getchar();  // Consume newline character
        fgets(logs[totalLogs].maintenance.emergency.emergencyType,
sizeof(logs[totalLogs].maintenance.emergency.emergencyType), stdin);
        logs[totalLogs].maintenance.emergency.emergencyType[strcspn(logs[totalLogs].maintenance.emerg
ency.emergencyType, "\n")] = '\0';
```

```
        printf("Enter emergency severity level: ");
        fgets(logs[totalLogs].maintenance.emergency.severityLevel,
sizeof(logs[totalLogs].maintenance.emergency.severityLevel), stdin);
        logs[totalLogs].maintenance.emergency.severityLevel[strcspn(logs[totalLogs].maintenance.emergen
cy.severityLevel, "\n")] = '\0';
    } else {
        printf("Invalid maintenance type\n");
        return;
    }

    totalLogs++;
}

void displayMaintenanceLogs(MaintenanceLog *logs) {
    printf("\nMaintenance Logs:\n");
    for (int i = 0; i < totalLogs; i++) {
        printf("\nLog ID: %d\n", logs[i].logID);
        printf("Aircraft ID: %d\n", logs[i].aircraftID);
        printf("Date: %s\n", logs[i].date);

        if (logs[i].maintenanceType == ROUTINE) {
            printf("Maintenance Type: Routine\n");
            printf("Duration: %d hours %d minutes\n", logs[i].maintenance.routine.duration.hours,
                                        logs[i].maintenance.routine.duration.minutes);
            printf("Description: %s\n", logs[i].maintenance.routine.description);
        } else if (logs[i].maintenanceType == EMERGENCY) {
            printf("Maintenance Type: Emergency\n");
            printf("Emergency Type: %s\n", logs[i].maintenance.emergency.emergencyType);
            printf("Severity Level: %s\n", logs[i].maintenance.emergency.severityLevel);
```

```
        }
    }
}

int main() {
    MaintenanceLog logs[MAX_LOGS];
    int user_input;
    bool is_on = true;

    while (is_on) {
        printf("\nAircraft Maintenance Tracker\n");
        printf("1. Add Maintenance Log\n2. Display Maintenance Logs\n3. Exit\n");
        printf("Enter your option: ");
        scanf("%d", &user_input);

        switch (user_input) {
            case 1:
                addMaintenanceLog(logs);
                break;
            case 2:
                displayMaintenanceLogs(logs);
                break;
            case 3:
                is_on = false;
                break;
            default:
                printf("Invalid choice! Please try again.\n");
                break;
        }
```

```
    }

    return 0;
}
```

5.spacecraft navigation system

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

#define MANUAL 1
#define AUTOMATIC 2
#define MAX_UPDATES 100

typedef struct {
    float x;
    float y;
    float z;
} Position;

typedef struct {
    float x;
    float y;
    float z;
} Velocity;

typedef struct {
    float speed;
    char direction[30];
} ManualMode;
```

```
typedef struct {
    int GPSCoordinates[3];  // Latitude, Longitude, Altitude
} AutomaticMode;

typedef union {
    ManualMode manual;
    AutomaticMode automatic;
} NavigationMode;

typedef struct {
    Position position;
    Velocity velocity;
    NavigationMode mode;
    int navigationMode;  // 1 for Manual, 2 for Automatic
} NavigationData;

static int totalUpdates = 0;

void updateNavigationData(NavigationData *data) {
    printf("Enter position (x, y, z): ");
    scanf("%f %f %f", &data->position.x, &data->position.y, &data->position.z);

    printf("Enter velocity (x, y, z): ");
    scanf("%f %f %f", &data->velocity.x, &data->velocity.y, &data->velocity.z);

    printf("Select navigation mode (1 for Manual, 2 for Automatic): ");
    scanf("%d", &data->navigationMode);
```

```
    if (data->navigationMode == MANUAL) {
        printf("Enter speed: ");
        scanf("%f", &data->mode.manual.speed);
        printf("Enter direction: ");
        getchar();  // To consume the newline left by previous input
        fgets(data->mode.manual.direction, sizeof(data->mode.manual.direction), stdin);
        data->mode.manual.direction[strcspn(data->mode.manual.direction, "\n")] = '\0';
    } else if (data->navigationMode == AUTOMATIC) {
        printf("Enter GPS coordinates (Latitude, Longitude, Altitude): ");
        scanf("%d %d %d", &data->mode.automatic.GPSCoordinates[0],
                    &data->mode.automatic.GPSCoordinates[1],
                    &data->mode.automatic.GPSCoordinates[2]);
    } else {
        printf("Invalid navigation mode!\n");
        return;
    }

    totalUpdates++;
}

void displayNavigationData(const NavigationData *data) {
    printf("\nCurrent Navigation Status:\n");
    printf("Position: (%.2f, %.2f, %.2f)\n", data->position.x, data->position.y, data->position.z);
    printf("Velocity: (%.2f, %.2f, %.2f)\n", data->velocity.x, data->velocity.y, data->velocity.z);

    if (data->navigationMode == MANUAL) {
        printf("Navigation Mode: Manual\n");
        printf("Speed: %.2f\n", data->mode.manual.speed);
        printf("Direction: %s\n", data->mode.manual.direction);
```

```
    } else if (data->navigationMode == AUTOMATIC) {
        printf("Navigation Mode: Automatic\n");
        printf("GPS Coordinates: (%d, %d, %d)\n",
               data->mode.automatic.GPSCoordinates[0],
               data->mode.automatic.GPSCoordinates[1],
               data->mode.automatic.GPSCoordinates[2]);
    }
}

int main() {
    NavigationData data[MAX_UPDATES];
    int user_input;
    int running = 1;

    while (running) {
        printf("\nSpacecraft Navigation System\n");
        printf("1. Update Navigation Data\n2. Display Current Status\n3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &user_input);

        switch (user_input) {
            case 1:
                if (totalUpdates < MAX_UPDATES) {
                    updateNavigationData(&data[totalUpdates]);
                } else {
                    printf("Maximum updates reached!\n");
                }
                break;
            case 2:
```

```
            for (int i = 0; i < totalUpdates; i++) {
                displayNavigationData(&data[i]);
            }
            break;
        case 3:
            running = 0;
            break;
        default:
            printf("Invalid choice, please try again.\n");
            break;
    }
  }

  return 0;
}
```

6.Problem 6: Flight Simulation Control

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

#define MAX_SIMULATIONS 50
#define MAX_AIRCRAFT_MODEL_LEN 30

typedef struct {
  int simulationID;
  char aircraftModel[MAX_AIRCRAFT_MODEL_LEN];
  float duration;
  union {
    struct {
```

```
            int throttle;
            int altitude;
        } manual;
        struct {
            int autopilotStatus;
            int autopilotAltitude;
        } automated;
    } controlSettings;
    int controlType;
} Simulation;

static int totalSimulations = 0;

void startSimulation(Simulation *sim) {
    printf("Enter simulation ID: ");
    scanf("%d", &sim->simulationID);

    printf("Enter aircraft model: ");
    getchar();  // Consume newline from previous input
    fgets(sim->aircraftModel, sizeof(sim->aircraftModel), stdin);
    sim->aircraftModel[strcspn(sim->aircraftModel, "\n")] = '\0'; // Remove newline character

    printf("Enter simulation duration (in hours): ");
    scanf("%f", &sim->duration);

    printf("Select control settings (1 for Manual, 2 for Automated): ");
    scanf("%d", &sim->controlType);

    if (sim->controlType == 1) {
```

```
        printf("Enter throttle level (0-100): ");
        scanf("%d", &sim->controlSettings.manual.throttle);
        printf("Enter altitude in feet: ");
        scanf("%d", &sim->controlSettings.manual.altitude);
    } else if (sim->controlType == 2) {
        printf("Enter autopilot status (0 for off, 1 for on): ");
        scanf("%d", &sim->controlSettings.automated.autopilotStatus);
        printf("Enter autopilot altitude in feet: ");
        scanf("%d", &sim->controlSettings.automated.autopilotAltitude);
    } else {
        printf("Invalid control setting!\n");
    }

    totalSimulations++;
}

void displaySimulationResults(const Simulation *sim) {
    printf("Simulation ID: %d\n", sim->simulationID);
    printf("Aircraft Model: %s\n", sim->aircraftModel);
    printf("Simulation Duration: %.2f hours\n", sim->duration);

    if (sim->controlType == 1) {
        printf("Control Setting: Manual\n");
        printf("Throttle: %d\n", sim->controlSettings.manual.throttle);
        printf("Altitude: %d feet\n", sim->controlSettings.manual.altitude);
    } else if (sim->controlType == 2) {
        printf("Control Setting: Automated\n");
        printf("Autopilot Status: %s\n", sim->controlSettings.automated.autopilotStatus ? "On" : "Off");
        printf("Autopilot Altitude: %d feet\n", sim->controlSettings.automated.autopilotAltitude);
```

```
    }
}

int main() {
    Simulation simulations[MAX_SIMULATIONS];
    int userInput;
    int isRunning = 1;

    while (isRunning) {
        printf("\n1. Start Simulation\n2. Display Simulation Results\n3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &userInput);

        switch (userInput) {
            case 1:
                if (totalSimulations < MAX_SIMULATIONS) {
                    startSimulation(&simulations[totalSimulations]);
                } else {
                    printf("Max simulations reached.\n");
                }
                break;
            case 2:
                if (totalSimulations > 0) {
                    displaySimulationResults(&simulations[totalSimulations - 1]);
                } else {
                    printf("No simulations available.\n");
                }
                break;
            case 3:
```

```
            isRunning = 0;
            break;
        default:
            printf("Invalid choice! Please try again.\n");
            break;
    }
  }

  return 0;
}
```

7.Aerospace Component Testing

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

#define MAX_TESTS 100

typedef struct {
  float pressure;
  float temperature;
} PhysicalTestData;

typedef struct {
  char softwareVersion[20];
  int bugsFound;
} SoftwareTestData;

typedef union {
```

```
    PhysicalTestData physical;
    SoftwareTestData software;
} TestData;

typedef struct {
    int testID;
    char componentName[30];
    TestData testData;
    int testType; // 1 for physical, 2 for software
} ComponentTest;

static int totalTests = 0;

void recordTestResult(ComponentTest *test) {
    printf("Enter component name: ");
    getchar();
    fgets(test->componentName, sizeof(test->componentName), stdin);
    test->componentName[strcspn(test->componentName, "\n")] = '\0';

    printf("Enter test type (1 for Physical, 2 for Software): ");
    scanf("%d", &test->testType);

    if (test->testType == 1) {
        printf("Enter pressure: ");
        scanf("%f", &test->testData.physical.pressure);
        printf("Enter temperature: ");
        scanf("%f", &test->testData.physical.temperature);
    } else if (test->testType == 2) {
        printf("Enter software version: ");
```

```
        scanf("%s", test->testData.software.softwareVersion);
        printf("Enter number of bugs found: ");
        scanf("%d", &test->testData.software.bugsFound);
    } else {
        printf("Invalid test type!\n");
        return;
    }

    totalTests++;
}

void displayTestSummary(const ComponentTest *test) {
    printf("Test ID: %d\n", test->testID);
    printf("Component Name: %s\n", test->componentName);

    if (test->testType == 1) {
        printf("Test Type: Physical\n");
        printf("Pressure: %.2f\n", test->testData.physical.pressure);
        printf("Temperature: %.2f\n", test->testData.physical.temperature);
    } else if (test->testType == 2) {
        printf("Test Type: Software\n");
        printf("Software Version: %s\n", test->testData.software.softwareVersion);
        printf("Bugs Found: %d\n", test->testData.software.bugsFound);
    }
}

int main() {
    ComponentTest tests[MAX_TESTS];
    int userInput;
```

```
int isRunning = 1;

while (isRunning) {
    printf("\n1. Record Test Result\n2. Display Test Summary\n3. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &userInput);

    switch (userInput) {
        case 1:
            if (totalTests < MAX_TESTS) {
                recordTestResult(&tests[totalTests]);
            }
            break;
        case 2:
            if (totalTests > 0) {
                displayTestSummary(&tests[totalTests - 1]);
            } else {
                printf("No test records available!\n");
            }
            break;
        case 3:
            isRunning = 0;
            break;
        default:
            printf("Invalid choice! Please try again.\n");
            break;
    }
}
```

```
    return 0;
}
```

8.crew management

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

#define MAX_CREW 50

typedef struct {
    int crewID;
    char name[30];
    char role[20];
    union {
        struct {
            char engineeringField[30];
            int experienceYears;
        } engineer;
        struct {
            char researchField[30];
            int publications;
        } scientist;
    } roleSpecificDetails;
} CrewMember;

static int totalCrewMembers = 0;

void addCrewMember(CrewMember *crew) {
    printf("Enter crew member ID: ");
```

```
    scanf("%d", &crew->crewID);

    printf("Enter crew member name: ");
    getchar();
    fgets(crew->name, sizeof(crew->name), stdin);
    crew->name[strcspn(crew->name, "\n")] = '\0';

    printf("Enter role (Engineer/Scientist): ");
    scanf("%s", crew->role);

    if (strcmp(crew->role, "Engineer") == 0) {
        printf("Enter engineering field: ");
        scanf("%s", crew->roleSpecificDetails.engineer.engineeringField);
        printf("Enter years of experience: ");
        scanf("%d", &crew->roleSpecificDetails.engineer.experienceYears);
    } else if (strcmp(crew->role, "Scientist") == 0) {
        printf("Enter research field: ");
        scanf("%s", crew->roleSpecificDetails.scientist.researchField);
        printf("Enter number of publications: ");
        scanf("%d", &crew->roleSpecificDetails.scientist.publications);
    } else {
        printf("Invalid role!\n");
        return;
    }

    totalCrewMembers++;
}

void displayCrewMember(const CrewMember *crew) {
```

```
    printf("Crew ID: %d\n", crew->crewID);
    printf("Name: %s\n", crew->name);
    printf("Role: %s\n", crew->role);

    if (strcmp(crew->role, "Engineer") == 0) {
        printf("Engineering Field: %s\n", crew->roleSpecificDetails.engineer.engineeringField);
        printf("Years of Experience: %d\n", crew->roleSpecificDetails.engineer.experienceYears);
    } else if (strcmp(crew->role, "Scientist") == 0) {
        printf("Research Field: %s\n", crew->roleSpecificDetails.scientist.researchField);
        printf("Publications: %d\n", crew->roleSpecificDetails.scientist.publications);
    }
}

int main() {
    CrewMember crew[MAX_CREW];
    int userInput;
    int isRunning = 1;

    while (isRunning) {
        printf("\n1. Add Crew Member\n2. Display Crew Member\n3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &userInput);

        switch (userInput) {
            case 1:
                if (totalCrewMembers < MAX_CREW) {
                    addCrewMember(&crew[totalCrewMembers]);
                }
                break;
```

```
            case 2:
                if (totalCrewMembers > 0) {
                    displayCrewMember(&crew[totalCrewMembers - 1]);
                } else {
                    printf("No crew members available!\n");
                }
                break;
            case 3:
                isRunning = 0;
                break;
            default:
                printf("Invalid choice! Please try again.\n");
                break;
        }
    }

    return 0;
}
```

9.Aerospace research data analysis

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

#define MAX_EXPERIMENTS 100

typedef struct {
    float numericalData[10];
    int dataCount;
} NumericalData;
```

```
typedef struct {
    char qualitativeData[5][100];
    int dataCount;
} QualitativeData;

typedef union {
    NumericalData numerical;
    QualitativeData qualitative;
} ResearchData;

typedef struct {
    int experimentID;
    char description[100];
    ResearchData data;
    int dataType; // 1 for numerical, 2 for qualitative
} Experiment;

static int totalExperiments = 0;

void analyzeResearchData(Experiment *data) {
    printf("Enter experiment ID: ");
    scanf("%d", &data->experimentID);

    printf("Enter experiment description: ");
    getchar();
    fgets(data->description, sizeof(data->description), stdin);
    data->description[strcspn(data->description, "\n")] = '\0';
```

```
    printf("Enter data type (1 for Numerical, 2 for Qualitative): ");
    scanf("%d", &data->dataType);

    if (data->dataType == 1) {
        printf("Enter number of numerical data points: ");
        scanf("%d", &data->data.numerical.dataCount);
        for (int i = 0; i < data->data.numerical.dataCount; i++) {
            printf("Enter numerical data point %d: ", i + 1);
            scanf("%f", &data->data.numerical.numericalData[i]);
        }
    } else if (data->dataType == 2) {
        printf("Enter number of qualitative data points: ");
        scanf("%d", &data->data.qualitative.dataCount);
        for (int i = 0; i < data->data.qualitative.dataCount; i++) {
            printf("Enter qualitative data point %d: ", i + 1);
            getchar();
            fgets(data->data.qualitative.qualitativeData[i], 100, stdin);
            data->data.qualitative.qualitativeData[i][strcspn(data->data.qualitative.qualitativeData[i], "\n")]
= '\0';
        }
    } else {
        printf("Invalid data type!\n");
        return;
    }

    totalExperiments++;
}

void displayResearchReport(const Experiment *data) {
```

```
    printf("Experiment ID: %d\n", data->experimentID);
    printf("Description: %s\n", data->description);

    if (data->dataType == 1) {
        printf("Data Type: Numerical\n");
        for (int i = 0; i < data->data.numerical.dataCount; i++) {
            printf("Numerical Data %d: %.2f\n", i + 1, data->data.numerical.numericalData[i]);
        }
    } else if (data->dataType == 2) {
        printf("Data Type: Qualitative\n");
        for (int i = 0; i < data->data.qualitative.dataCount; i++) {
            printf("Qualitative Data %d: %s\n", i + 1, data->data.qualitative.qualitativeData[i]);
        }
    }
}

int main() {
    Experiment experiments[MAX_EXPERIMENTS];
    int userInput;
    int isRunning = 1;

    while (isRunning) {
        printf("\n1. Analyze Research Data\n2. Display Research Report\n3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &userInput);

        switch (userInput) {
            case 1:
                if (totalExperiments < MAX_EXPERIMENTS) {
```

```
            analyzeResearchData(&experiments[totalExperiments]);
        }
        break;
    case 2:
        if (totalExperiments > 0) {
            displayResearchReport(&experiments[totalExperiments - 1]);
        } else {
            printf("No experiments available!\n");
        }
        break;
    case 3:
        isRunning = 0;
        break;
    default:
        printf("Invalid choice! Please try again.\n");
        break;
  }
 }

 return 0;
}
```

10.Rocket launch scheduler

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

#define MAX_LAUNCHES 50

typedef struct {
```

```
    int launchID;
    char rocketName[30];
    char date[20];
    union {
        struct {
            char status[20];
            int countdown;
        } scheduled;
        struct {
            char status[20];
            char completionDate[20];
        } completed;
    } status;
    int statusType; // 1 for scheduled, 2 for completed
} Launch;

static int totalLaunches = 0;

void scheduleLaunch(Launch *launch) {
    printf("Enter launch ID: ");
    scanf("%d", &launch->launchID);

    printf("Enter rocket name: ");
    getchar();
    fgets(launch->rocket.
```