Design and implement a program in C to manage details about researchers and their associated projects. The program must meet the following requirements:Researcher Structure:
Create a structure to store the following details:
researcherID (integer): Unique ID for the researcher.
name (string): Name of the researcher.
age (integer): Age of the researcher.
department (string): Department or research area.
yearsOfExperience (float): Number of years the researcher has been active.
Project Details:
Use a union to represent the project details:
grantAmount (float): Funding received for the project.
publicationCount (integer): Number of publications generated from the project.
Use an enumeration to classify projects into types:
FUNDED, SELF_FINANCED, or COLLABORATIVE.
Dynamic Memory Allocation:
Allow the program to allocate memory dynamically for storing details of researchers based on user input.
Features to Implement:
Input Researcher Details: Allow the user to input researcher details and their associated project.
Display All Researchers: Display the details of all researchers, including their project information.
Search Researcher: Search for a researcher by their researcherID or name.
Sort Researchers:
By years of experience in descending order.Alphabetically by department.
Calculate Statistics:
Average years of experience across all researchers.
Total funding received for funded projects.
Update Project Information: Update project details (grant amount and publication count) for a specific researcher using their ID.
Exit: Free all allocated memory and exit the program.
Use functions to implement modularity (e.g., input, display, search, sort, etc.).
Handle invalid inputs gracefully and provide error messages when appropriate.
Ensure the program is efficient and uses memory optimall
1. Input Researcher Details
2. Display All Researchers
3. Search Researcher by ID or Name
4. Sort Researchers by Experience or Department
5. Calculate and Display Researcher and Project Statistics
6. Update Project Information for a Researcher

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

```
typedef struct {
   int researcherID;
   char name[50];
   int age;
   char department[50];
   float yearsOfExperience;
   float grantAmount;
   int publicationCount;
} Researcher;


void inputResearcher(Researcher **research, int *count);
void displayResearchers(Researcher *research, int count);
void searchResearcher(Researcher *research, int count);
void sortResearchersByExperience(Researcher *research, int count);
void calculateAndDisplayStatistics(Researcher *research, int count);
void updateProjectInformation(Researcher *research, int count);


int main() {
   Researcher *research = NULL;
   int count = 0;
   int choice;

   do {
      printf("\nResearcher Management System\n");
      printf("1. Input Researcher Details\n");
      printf("2. Display All Researchers\n");
      printf("3. Search Researcher by ID\n");
      printf("4. Sort Researchers by Experience\n");
      printf("5. Calculate and Display Statistics\n");
      printf("6. Update Project Information for a Researcher\n");
      printf("7. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);

      switch (choice) {
         case 1:
            inputResearcher(&research, &count);
            break;
         case 2:
            displayResearchers(research, count);
            break;
         case 3:
            searchResearcher(research, count);
            break;
```

```
        case 4:
            sortResearchersByExperience(research, count);
            break;

        case 5:
            calculateAndDisplayStatistics(research, count);
            break;
        case 6:
            updateProjectInformation(research, count);
            break;
        case 7:
            printf("Exiting the system.\n");
            free(research);
            break;
        default:
            printf("Invalid choice. Please try again.\n");
    }
  } while (choice != 7);

  return 0;
}


void inputResearcher(Researcher **research, int *count) {
  *research = realloc(*research, (*count + 1) * sizeof(Researcher));
  if (*research == NULL) {
    printf("Memory allocation failed.\n");
    return;
  }

  printf("Enter Researcher ID: ");
  scanf("%d", &(*research)[*count].researcherID);
  printf("Enter Name: ");
  scanf("%s", (*research)[*count].name);
  printf("Enter Age: ");
  scanf("%d", &(*research)[*count].age);
  printf("Enter Department: ");
  scanf("%s", (*research)[*count].department);
  printf("Enter Years of Experience: ");
  scanf("%f", &(*research)[*count].yearsOfExperience);

  (*research)[*count].grantAmount = 0;
  (*research)[*count].publicationCount = 0;

  (*count)++;
  printf("Researcher details added successfully.\n");
}
```

```
void displayResearchers(Researcher *research, int count) {
  if (count == 0) {
    printf("No researchers found.\n");
    return;
  }

  printf("\n--- Researcher Details ---\n");
  for (int i = 0; i < count; i++) {
    printf("ID: %d\n", research[i].researcherID);
    printf("Name: %s\n", research[i].name);
    printf("Age: %d\n", research[i].age);
    printf("Department: %s\n", research[i].department);
    printf("Years of Experience: %.2f\n", research[i].yearsOfExperience);
    printf("Grant Amount: %.f\n", research[i].grantAmount);
    printf("Publication Count: %d\n\n", research[i].publicationCount);
  }
}


void searchResearcher(Researcher *research, int count) {
  if (count == 0) {
    printf("No researchers found.\n");
    return;
  }

  int id;
  printf("Enter Researcher ID to search: ");
  scanf("%d", &id);

  for (int i = 0; i < count; i++) {
    if (research[i].researcherID == id) {
      printf("\nResearcher Found:\n");
      printf("ID: %d\n", research[i].researcherID);
      printf("Name: %s\n", research[i].name);
      printf("Age: %d\n", research[i].age);
      printf("Department: %s\n", research[i].department);
      printf("Years of Experience: %.2f\n", research[i].yearsOfExperience);
      printf("Grant Amount: %.2f\n", research[i].grantAmount);
      printf("Publication Count: %d\n", research[i].publicationCount);
      return;
    }
  }

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

```
void calculateAndDisplayStatistics(Researcher *research, int count) {
  if (count == 0) {
    printf("No researchers found.\n");
    return;
  }

  float totalExperience = 0;
  float totalFunding = 0;

  for (int i = 0; i < count; i++) {
    totalExperience += research[i].yearsOfExperience;
    totalFunding += research[i].grantAmount;
  }

  printf("\nStatistics\n");
  printf("Average Years of Experience: %.2f\n", totalExperience / count);
  printf("Total Grant Funding: %.2f\n", totalFunding);
}


void updateProjectInformation(Researcher *research, int count) {
  if (count == 0) {
    printf("No researchers found.\n");
    return;
  }

  int id;
  printf("Enter Researcher ID to update project information: ");
  scanf("%d", &id);

  for (int i = 0; i < count; i++) {
    if (research[i].researcherID == id) {
      printf("Enter Grant Amount: ");
      scanf("%f", &research[i].grantAmount);
      printf("Enter Publication Count: ");
      scanf("%d", &research[i].publicationCount);
      printf("Project information updated successfully.\n");
      return;
    }
  }

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

```
void sortResearchersByExperience(Researcher *research, int count) {
   for (int i = 0; i < count - 1; i++) {
      for (int j = 0; j < count - i - 1; j++) {
         if (research[j].yearsOfExperience < research[j + 1].yearsOfExperience) {
            Researcher temp = research[j];
            research[j] = research[j + 1];
            research[j + 1] = temp;
         }
      }
   }
   printf("Researchers sorted by experience.\n");
}
```