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

typedef struct person
{
 int Student_ID;
 char First_Name[50];
 char Last_Name[50];
 float Exam_1;
 float Exam_2;
 float Final_Exam;
 float Total_Score;
} per1, per2, per3;

void Add_Records(struct person*, int n);
void disp_records(struct person*, int n);
void Tot_Score(struct person*, int n);
void Find_Student(struct person*, int n);
void Max(struct person *pers, int n);
void Min(struct person *pers, int n);
void Sort(struct person *pers, int n);

void Sort(struct person *pers, int n)
{
  int t;
 int r;
 int x;
 float *temp; //Total Score
 int temp_student_id;

 int count;
 for(r = 0; r < n; r++)
 {
   for(t = 0; t < n; t++)
   {
     if(pers[r].Total_Score > pers[t].Total_Score)
     {
       //Total Score
       *temp = pers[r].Total_Score;
       pers[r].Total_Score = pers[t].Total_Score;
       pers[t].Total_Score = *temp;

       //Student ID
       //only type def structures can use the "=" operator
       temp_student_id = pers[t].Student_ID;
       pers[t].Student_ID = pers[r].Student_ID;
       pers[r].Student_ID = temp_student_id;
     }
   }
 }
 printf("Scores ranging from the highest and goint toward the lowest");
 for(t = 0; t < n; t++)
 {
   printf("\nTotal Score: %f", pers[t].Total_Score); // Nice.
   printf("\nStudent ID: %d", pers[t].Student_ID);
 }
}

void Add_Records(struct person *pers, int n)
{

 for(int i = 0; i < n; i++)
 {
   printf("\t\t\t_________________________\n");/* this print stmt is for
formatting / UI.*/

   printf("\t\t\tStudent ID:\t\t");
```

```
   scanf("%d", &pers[i].Student_ID);

   printf("\t\t\tFirst Name:\t\t");
   scanf("%s", pers[i].First_Name);

   printf("\t\t\tLast Name:\t\t");
   scanf("%s", pers[i].Last_Name);

   printf("\t\t\tExam One:\t\t");
   scanf("%f", &pers[i].Exam_1);

   printf("\t\t\tExam Two:\t\t");
   scanf("%f", &pers[i].Exam_2);

   printf("\t\t\tFinal Exam:\t\t");
   scanf("%f", &pers[i].Final_Exam);

   Tot_Score(pers, n);
   printf("\t\t\tTotal Score:\t%f\n\n", pers[i].Total_Score);
   //printf("\t\t\t_________________________\n");

 }
}

void disp_records(struct person *pers, int n)
{
 for(int i = 0; i < n; i++)
 {
   printf("\t\t\t_________________________\n");
   printf("\t\t\tStudent ID: \t%d\n", pers[i].Student_ID);
   printf("\t\t\tFirst Name: \t%s\n", pers[i].First_Name);
   printf("\t\t\tLast Name: \t\t%s\n", pers[i].Last_Name);
   printf("\t\t\tExam One: \t\t%f\n", pers[i].Exam_1);
   printf("\t\t\tExam Two: \t\t%f\n", pers[i].Exam_2);
```

```c
    printf("\t\t\tFinal Exam: \t%f\n", pers[i].Final_Exam);
    printf("\t\t\tFinal Score: \t%f\n", pers[i].Total_Score);
  }
}

void Tot_Score(struct person *pers, int n)
{
 for(int i = 0; i < n; i ++)
 {
   pers[i].Total_Score = pers[i].Exam_1 + pers[i].Exam_2
+pers[i].Final_Exam;
 }
}

void Find_Student(struct person *pers, int n)
{
 int i;
 printf("\t\t\tStudent # (Don't confuse with Student ID): "); /* "#"
refers to indexing number.*/
 scanf("%d", &i);

 printf("\t\t\t_________________________\n");
 printf("\t\t\tStudent ID: \t%d\t \n", pers[i].Student_ID);
 printf("\t\t\tFirst Name: \t%s \n", pers[i].First_Name);
 printf("\t\t\tLast Name: \t\t%s \n", pers[i].Last_Name);
 printf("\t\t\tExam One: \t\t%f \n", pers[i].Exam_1);
 printf("\t\t\tExam Two: \t\t%f \n", pers[i].Exam_2);
 printf("\t\t\tFinal Exam: \t%f \n", pers[i].Final_Exam);
 printf("\t\t\tTotal Score: \t%f \n", pers[i].Total_Score);
}

void Max(struct person *pers, int n)
{
 int i;
 int l;
 int count = 0;
 float maxman = pers[0].Total_Score;
 for(i = 0; i < n; i++)
 {
   if(pers[i].Total_Score > maxman)
   {
     maxman = pers[i].Total_Score;
     count ++;
   }
   else
   {
     // To handle any repetition in high scores.
   }
   /*else(pers[i].Total_Score = maxman);
   {
     // To handle any repetition in high scores.
     printf("\n\t\t\t_________________________\n");
     printf("There are multiple highest scores\n");
     printf("\t\t\tStudent ID: \t%d\n", pers[count].Student_ID);
     printf("\t\t\tFirst Name: \t%s\n", pers[count].First_Name);
     printf("\t\t\tLast Name: \t\t%s\n", pers[count].Last_Name);
     printf("\t\t\tTotal Score: \t%f\n", pers[count].Total_Score);
   }*/

 }
 printf("\t\t\t_________________________\n");
 printf("\t\t\tStudent ID: \t%d\n", pers[count].Student_ID);
 printf("\t\t\tFirst Name: \t%s\n", pers[count].First_Name);
 printf("\t\t\tLast Name: \t\t%s\n", pers[count].Last_Name);
 printf("\t\t\tTotal Score: \t%f\n", pers[count].Total_Score);
}

void Min(struct person *pers, int n)
```

```
{
 int i;
 int count = 0;
 float maxman = pers[0].Total_Score;
 for(i = 0; i < n; i++)
 {
   if(pers[i].Total_Score < maxman)
   {
     maxman = pers[i].Total_Score;
     count ++;
   }
 }
 printf("\t\t\t_________________________\n");
 printf("\t\t\tStudent ID: \t%d\n", pers[count].Student_ID);
 printf("\t\t\tFirst Name: \t%s\n", pers[count].First_Name);
 printf("\t\t\tLast Name: \t\t%s\n", pers[count].Last_Name);
 printf("\t\t\tTotal Score: \t%f\n", pers[count].Total_Score);
}

int main (void) {
  int i; // if statement parameter    --> (Use: Menu Functionality)
 int j = 1; // while loop parameter  --> (Use: Menu Functionality)
 int kinput; // array input          --> (Use: Array Input)
 int y;

 int n;
 struct person per[n];
 struct person *pers;
 pers = (struct person*)malloc(n * sizeof(struct person));


 printf("\nType a number between 1 and 7 to navigate the menu.\n");
 printf("1.) Add Student Records \n2.) View all Student's Records \n3.)
View a Students Records \n4.) Student With Max Score (First, Last Name)
```

```
\n5.) Student Who Has the Min Score (First and Last Name) \n6.) View
Records Sorted by Total Score.  \n7.) Quit the Program \n\n");


 while (j == 1) {
   printf("\nType a number between 1 & 7 to navigate the menu:   --> \t");
   scanf("%d", &y);
   if (y >= 1 && y <= 7 ) {
     switch (y) {


       // Below 1.) Add Student Records
       case (1):
         printf("\tType the number of students\t");
         scanf("%d", &n);
         printf("\t%d.) Add Student Records\n", y);
         Add_Records(pers, n);
         //printf("%lu", sizeof(per1));
         break;



       // Below 2.) View all Student's Records
       case (2):
         printf("\t%d.) Display Student Records:\n", y);
         disp_records(pers, n);
         break;

       // Below 3.) View a Students Records
       case (3):
         printf("\t%d.) View a Students Records\n", y);
         Find_Student(pers, n);

         break;
```

```
        // Below 4.) Student With Max Score (First, Last Name)
        case (4):
          printf("\t%d.) Student With Max Score\n", y);
          Max(pers,n);
          break;

        // Below 5.) Student Who Has the Min Score (First and Last Name)
        case (5):
          printf("\t%d.) Student Who Has the Min Score\n", y);
          Min(pers, n);
          break;

        // Below 6.) View Records Sorted by Total Score.
        case (6):
          printf("\t%d.) View Records Sorted by Total Score\n", y);
          Sort(pers, n);
          break;



        // Below 7.) Exit Program.
        case (7):
          printf("\t%d.) Quit the Program\n", y);
          j = 0;
          break;



        }
      }
    else if (y>=8 || y<=0)
      // Below 8.) Error
      printf("\tE.) Error\n");

  }
  }
```