CS29003 ALGORITHMS LABORATORY
Divide and Conquer
Last Date of Submission: 6 – September – 2018

General Instruction

1. Please do not use any global variable unless you are explicitly instructed so.
2. Please use proper indentation in your code.
3. Please name your file as `<roll_no>_<assignment_no>`. For example, if your roll number is 14CS10001 and you are submitting assignment 3, then name your file as `14CS10001_3.c or 14CS10001_3.cpp` as applicable.
4. Please write your name and roll number at the beginning of your program.

There are n students in a school. Let x_i and y_i be the marks scored by student i in the Math and English examinations respectively. Each x_i and y_i are integers between 0 and 100, both inclusive. Let us call (x_i, y_i) the *score profile* of student i. The school wants to offer assistance to the students based on their scores, and wants to find students with similar performances. To begin with, a natural first step for the school is to identify two students whose score profiles are closest. As a notion of closeness (or rather farness) of two score profiles (x_1, y_1) and (x_2, y_2), the school uses the L_1 distance of the pair defined as follows: $d((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|$.

PART-I Define the following structure type for storing the score profiles of students:

```
typedef struct score{
        int mscore;
        int escore;
}score, *scorelist;
```

Write a divide and conquer algorithm for finding a closest pair of score tuples. Write a function **NearesProfiles()** with the following prototype:

int NearestProfiles(scorelist T, int n, score *score1, score *score2);

where T is an array of n score profiles. **NearestProfiles()** finds a pair of score profiles with minimum L_1-distance (defined above), stores the score profiles in structures pointed to by score1 and score2, and returns the L_1-distance between them. **NearestProfiles()** must run in worst case $O(n \log n)$ time (n is the number of score profiles in T).

PART-II The school authority wants to compare students belonging to two different sections. Using divide and conquer, write a function with the following prototype to find out a nearest pair of score profiles from two different sections, as well as the L_1 distance between them.

int ClusterDist(scorelist section1, int n, scorelist section2, int m, score *score1, score *score2).

Arrays section1 and section2 contain the score profiles of the two sections. The number of score profiles in the two sections are n and m respectively. The nearest pair of score profiles are stored in locations indicated by score1 and score2. The function returns the L_1 distance between them.

main()
In **main()** do the following:

1. Make the following structure declarations:
   score profile1, profile2;
2. Take as input the number of students n.
3. Take as input the score profiles of n students of the first section.
4. Call **NearestProfiles()**. Print the pair of scores found, and the distance between them.
5. Take as input an integer m.
6. Take as input the score profiles of m students of the second section.
7. Call **ClusterDist()**. Then print the nearesr pair of score profiles from different sections and the distance between them.

Sample output: Enter no. of students: 10
Enter math and english scores: 56 78
Enter math and english scores: 48 99
Enter math and english scores: 54 72
Enter math and english scores: 97 54
Enter math and english scores: 100 89
Enter math and english scores: 29 51
Enter math and english scores: 39 44
Enter math and english scores: 96 99
Enter math and english scores: 88 87
Enter math and english scores: 41 37

closest pair:(54,72) and (56, 78)
distance=8

Enter the size of the second cluster:8
Enter math and english scores: 11 5
Enter math and english scores: 5 65
Enter math and english scores: 28 12
Enter math and english scores: 74 94
Enter math and english scores: 42 41
Enter math and english scores: 9 13
Enter math and english scores: 36 48
Enter math and english scores: 16 19

closest pair:(39,44) and (42, 41)
distance=6