int[] findMaximumDistance(int distanceOfCities[], int start, int end);

Jack would like to travel in his country, and he needs to know the distances from his city (Palo Alto) to other cities. Fortunately, he is given an array, distanceOfCities, that contains distance of cities from Palo Alto to other cities. You will implement an algorithm so that he will know the maximum distance between two cities.

Implement ***a recursive function*** *as defined above which takes an integer array and the start and end indices for the recursion of the function. The function will return an array of integers (which should be 2 integers) giving the maximum distance for the elements of the given array. Then you will print the elements of the returned array in your main function.*

You can follow the algorithm steps that are described below:

- *Base case 1: You have only one element in your array (start = end)*
- *Base case 2: You have two elements in your array (start + 1 = end)*
- *Search left part of the array by recursion*
    - ***findMaximumDistance(distanceOfCities, start, mid)***
- *Search right part of the array by recursion*
    - ***findMaximumDistance(distanceOfCities, mid+1, end)***
- *Check the elements returned from the left part (2 elements giving the maximum distance) and the right (2 elements giving the maximum distance) part (4 elements will be returned totally), then find and return two elements giving the maximum distance out of these 4 elements.*

distanceOfCities = {120, 200, 105, 300, 295}

The function should return {300,105} which provides maximum distance between the cities.

Hint: *For finding the maximum distance of elements in an array, you should find minimum and maximum elements of the array.*

Implement ***a recursive function*** *described below by following all the comments. The comments will make you reach the implementation target if you follow them very carefully.*

int recursiveMinStrDiff(char str1[], char str2[], int m, int n);

First let's try to cover the algorithm. You are given 2 strings and the length of the strings as parameters. Your function will return the minimum number of operations required to convert 'str1' into 'str2'.

You can follow the algorithm steps that are described below:

- Base case 1: If str1 is empty, it is needed to insert all characters of str2 into str1. It means that you should return the length of str2, when the length of the str1 is 0.
- Base case 2: If str2 is empty, it is needed to remove all characters from str1. It means that you should return the length of str1, when the length of the str2 is 0.
- Recursive case 1: If last characters of the strings are the same, since there is no such a thing to do, you ignore them and return the recurrence for the remaining characters by ***recursiveMinStrDiff(str1, str2, m-1, n-1).***
    - For example, "han" and "tan", "n" is the same for both, continue recurring for "ha" and "ta".
- Recursive case 2: If last characters are not the same, consider all three operations such as **Add, Remove, and Replace** on last character of str1, recursively compute minimum costs for all three operations and **take minimum of three values**.
    - ***Add: recursiveMinStrDiff(str1, str2, m, n-1)***
    - ***Remove : recursiveMinStrDiff(str1, str2, m -1 , n)***
    - ***Replace : recursiveMinStrDiff(str1, str2, m -1 , n - 1)***
    - ***You can return 1 + the minimum of three operations Add, Remove, Replace results.***

/* These strings as an example would return 2. Last four characters are the same. We basically need to convert "mo" to "su". This can be done using two operations. Replace 'o' with u', and replace 'm' with s' */

char str1[] = "monday"; char str2[] = "sunday";

/* These strings as an example would return 2. This can be done using two operations. Remove 'a', and replace 'l' with ' t' */

char str1[] = "tesla"; char str2[] = "test";

/* These strings as an example would return 3. Last three and first characters are same. We basically need to convert "un" to "atur". This can be done using below three operations. Replace 'n' with 'r', insert t, insert a */

char str1[] = "sunday";char str2[] = "saturday";

```
#include <string.h>

#define MIN(x, y) (((x) < (y)) ? (x) : (y))

#define THREE_MIN(a,b,c) MIN(MIN(a,b),c)

int main(){

   //printf( "%d", THREE_MIN(9,6,7));

   char str1[] = "tesla";

   char str2[] = "test";

   printf("%d", recursiveMinStrDiff (str1, str2, strlen(str1), strlen(str2)));

   return 0;

}
```