Lab Assignment #9

Purpose: Recursive Function
Collaboration: You have to work in a group of two.

Getting started

Change into your cse220 directory
Create a new directory called lab09
Change into the new directory
Implement the two programs below in your lab09 directory

IMPORTANT NOTE (Pre-requisite):

There are two ways to pass inputs to the functions, called ‘passing by value’ and ‘passing by reference’.

Pass by value: is when you only pass a value to a function as its input. The function can use/modify the value and create its outputs. Like passing an integer to a function.

Pass by reference: is when you pass the actual variable to a function and if the function modify the input, it will affect the original variable too and changes that value outside of the function too. One example is when you pass an array to function and if the function modify the elements of the input array, when you get out of the function and check the arrays input, they have been changed.

Both methods can be used according to the way of implementation. Here, in the merge sort, as you can see, there is no outputs for the functions because arrays will be called by reference by default. We will learn more in lectures soon.
Merge Sort

Write a program lab9.c as follows:

First, you need to have a main function as follows, this main function is for testing the merge sort function. You don't need to modify the main.

```c
/* Driver program to test the merge sort function */
int main()
{
    int arr[] = {12, 11, 13, 5, 6, 7, 14, 8};
    int arr_size = sizeof(arr)/sizeof(arr[0]);

    printf("Given array is \n");
    printArray(arr, arr_size);

    mergeSort(arr, 0, arr_size - 1);

    printf("\nSorted array is \n");
    printArray(arr, arr_size);
    return 0;
}
```

For simplicity, we defined a function called `printArray()` that prints all the elements of the array. The `printArray()` function's signature should be as follows. You need to implement the body of this function according to the provided inputs and output. (hint: this function only prints the array elements one by one separated by one whitespace and prints a new line (\n) at the end)

```c
/* UTILITY FUNCTIONS */
/* Function to print an array */
void printArray(int A[], int size)
{
}
```
Then you need to take care of the merge function. For defining merge sort, usually we define two functions. The first one is the recursive merge sort function that breaks up its input into two smaller arrays and recursively calls itself. Here are the general steps:

1- Your main() will call the mergesort() function with three inputs: an array and two integers. The array input is the array that we want to sort. Two inputs are two indices of ‘left’ and ‘right’ indicating the beginning and end of the part, respectively, that we want to sort. Inside the main(), since we want to sort the whole list, we call it as (look at the main function)

   mergeSort(arr, 0, arr_size - 1);

2- Then, inside the mergeSort():
   a. You should check if ‘left’ input is less than the ‘right’ input or not.
      i. If ‘left < right’, means the size of the array is at least two (WHY?), so you still can (need to) divide the array into two smaller arrays.
      ii. Then you break the array from middle and pass each new half to the mergeSort() again (calling recursive).
      iii. Then you need to merge the outputs of these two recursive calls that you made in previous step. To merge two arrays, you need to call another function called ‘merge()’. We will talk about it in the next step.
      iv. If ‘left < right’ is wrong, you don’t need to do anything.
   b. Here is the mergeSort() declaration and you need to complete the body based on the steps.

   /*************************************************************************
   mergeSort is for left index and right is right index of the sub-array
   of arr to be sorted */
   void mergeSort(int arr[], int left, int right)
   {
   
   
   }
3- Now the last function that you need to define is the merge function that one array and four integer indices of ‘left1’, ‘right1’, ‘left2’ and ‘right2’ indicating the beginning and end for two sub arrays that we try to merge. Here the assumption is that the two sub arrays are sorted and you want to merge them.

Here is the declaration of the merge() function and you need to implement the body accordingly.

```c
/* Function to merge the two halves arr[left1...right1] and arr[left2...right2] of array arr[] */
void merge(int arr[], int left1, int right1, int left2, int right2)
{

}
```

So at the end your code should be as follows completed with the body of the mentioned functions. 

*Show the TA your program and the results*
/* C program for merge sort */
#include<stdio.h>
#include<stdlib.h>

/* Function to merge the two halves arr[left1...right1] and arr[left2...right2] of array arr[] */
void merge(int arr[], int left1, int right1, int left2, int right2)
{

}

/* left is for left index and right is right index of the sub-array of arr to be sorted */
void mergeSort(int arr[], int left, int right)
{

}

/* UTILITY FUNCTIONS */
/* Function to print an array */
void printArray(int A[], int size)
{

}

/* Driver program to test the merge sort function */
int main()
{
    int arr[] = {12, 11, 13, 5, 6, 7, 14, 8};
    int arr_size = sizeof(arr)/sizeof(arr[0]);

    printf("Given array is \n");
    printArray(arr, arr_size);

    mergeSort(arr, 0, arr_size - 1);

    printf("\nSorted array is \n");
    printArray(arr, arr_size);
    return 0;
}