

## CSE108 – Computer Programming Laboratory

### Spring 2022, Lab 8

*This lab will be graded on a scale of 100. No collaboration is permitted. You have 80 mins.*

PS: You cannot use pointers.

Write a program that performs the following task. You need to write the comments if not, you get -5 for each function.

#### Part 1. (40 pts) Recursive Palindrome String Check:

Write a recursive function "void reverse(char str[], char reversed[], int i)" that takes 3 arguments. Original string, reversed string, and index.

In main, you can compare two strings then print to the terminal palindrome or not.

Hints:

- You don't need to take string from user.
- You may use '\0' for end of string.
- You may send the length of original string in main.
- You don't need to use strcpy etc., you can select char by char.
- You may write this function just use three-line code that one is if condition.
- These are just hints, not obligations.

#### Part 2. (60 pts) Merge Sort:

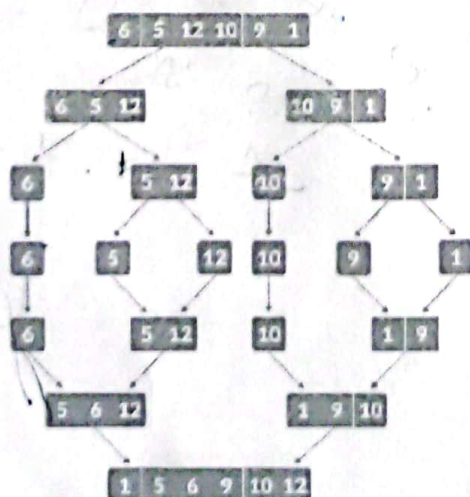
Write a recursive sort function "void mergeSort(int arr[], int l, int r)" that takes 3 arguments: Array, left index and right index. You can see the in Figure 1 how it works.

You need to use merge function for sorted subarrays that codes give you in next page.

Hints:

- No hints for this part 😊

Figure 1(How it works):



```

// Merge two subarrays L and M into arr
void merge(int arr[], int p, int q, int r) {

    // Create L ← A[p..q] and M ← A[q+1..r]
    int n1 = q - p + 1;
    int n2 = r - q;

    int L[n1], M[n2];

    for (int i = 0; i < n1; i++)
        L[i] = arr[p + i];
    for (int j = 0; j < n2; j++)
        M[j] = arr[q + 1 + j];

    // Maintain current index of sub-arrays and main array
    int i, j, k;
    i = 0;
    j = 0;
    k = p;

    // Until we reach either end of either L or M, pick larger among
    // elements L and M and place them in the correct position at A[p..r]
    while (i < n1 && j < n2) {
        if (L[i] <= M[j]) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = M[j];
            j++;
        }
        k++;
    }

    // When we run out of elements in either L or M,
    // pick up the remaining elements and put in A[p..r]
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2) {
        arr[k] = M[j];
        j++;
        k++;
    }
}

```