2) (10 pts) DSN (Sorting)

Complete the following merge function that is used as part of the merge sort process. The function performs merge operation from left to mid and mid+1 to right index of the array.

```
void merge(int arr[], int left, int mid, int right)
{
    int i, j, k;
    int n1 = mid - left + 1; //size of the left array
    int n2 =  right - mid; //size of the right array

    /* create temp arrays */
    int *L = (int*) malloc(n1*sizeof(int)); //left array
    int *R = (int*) malloc(n2*sizeof(int)); //right array

    /* Copy data to temp arrays L[] and R[] */
    for (i = 0; i < n1; i++)
        L[i] = arr[left + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[mid + 1+ j];

    /* Merge the temp arrays back into arr[l..r]*/
    i = 0; // Initial index of left subarray
    j = 0; // Initial index of right subarray
    k = left; // Initial index of merged subarray
    // Complete the remaining part of the code that will
    // merge L and R array into arr


    while(i<n1 || j< n2) {                   // Grading: 2 pts
        if( j==n2 || (i<n1 && L[i] < R[j])){ // Grading: 3 pts

            arr[k] = L[i];                   // Grading: 1 pt
            i++;                             // Grading: 1 pt
        }
        else {
            arr[k] = R[j];                   // Grading: 1 pt
            j++;                             // Grading: 1 pt
        }
        k++;                                 // Grading: 1 pt
    }
}
```

Grading Notes: Longer ways to do this…typical errors will involve AOOB or not copying the last items after one list is exhausted. (Stops before end && in loop = 8/10, Copy back all items into arr but wrong order, for example alternating, – 4/10, any AOOB but otherwise correct – 8/10)