**Task 1:**

The merge sort method is used by the code to count the number of inversions in an array. It splits the array into two smaller parts, counts the number of inversions as the two parts are put back together, and gives you the total number of inversions. When two items in a collection are out of order, this is called an inversion. In the example of how the code is used, the collection [5, 4, 3, 2, 1] has 10 inversions.

**Task 2:**

The code sets up two methods, "abs\_val" and "getmax," that can be used to find the largest sum in an array. 'abs\_val' uses a loop to find the largest absolute value in the list and return it. "getmax" cuts the group in half over and over again. It uses repeated calls to 'getmax' to figure out the largest amount in each half. It also figures out the largest sum that crosses the middle of the collection by finding the largest value in the left half and adding it to the square of the largest absolute value in the right half. The function gives back the largest sum from the left half, the right half, and the middle sum.

**Task 3:**

The split function picks a pivot element and rearranges the array so that all elements smaller than the pivot are on its left and all elements bigger than the pivot are on its right. It gives the number of the pivot after partitioning. The quick\_sort method divides the array into subarrays and runs the partition function on each one. It keeps doing this until each subarray is sorted on its own. In the example, the array [9, 5, 4, 6, 1, 3, 2, 9] is passed to quick\_sort with the starting index 0 and the ending index n. The function sorts the array and returns its sorted form.

**Task 4:**

Arr and k are the parameters for find\_Kth. It sets the starting index p to 0 and the ending index r to the array length minus 1. The function partitions the pivot, the last member of the array, in a while loop. It rearranges the array so that every element less than or equal to the pivot are on its left and all greater than it are on its right. After partitioning, the algorithm compares pivot\_idx to the desired k-th index. It returns the k-th smallest value if they are equal. To narrow the search range, it updates the ending index r to the pivot index minus 1. If the pivot index is smaller than the k-th index, it changes the starting index p to the pivot plus 1. The code then iterates through the queries, reads each query value from the input file, finds the k-th smallest value using find\_Kth, and writes the results to the output file.