After implementing and testing merge sort on different types of input data (random, decreasing, and increasing), we can see that the running time of merge sort is significantly faster than both selection sort and insertion sort, especially for larger input sizes.

When comparing the results of the three different cases, we can see that merge sort is the most efficient when sorting already sorted data, as it takes the least amount of time to sort this type of data. This is because merge sort is based on the divide and conquer approach, which means that it divides the input array into smaller sub-arrays, sorts them recursively, and then merges them to obtain the sorted output array. In the case of already sorted data, the divide and conquer approach is much more effective because there are fewer sub-arrays to sort and merge.

On the other hand, merge sort takes the longest to sort reverse sorted data. This is because in this case, each recursive call creates sub-arrays that are unbalanced in size, making the merging process more complex and time-consuming.

Overall, the results obtained from this experiment confirm that merge sort is an efficient sorting algorithm, especially for larger input sizes.