1. You are given:

N items, each with (value, weight) A knapsack with capacity W However, before applying the Fractional Knapsack, you must:

Sort items by: (Choose the most appropriate algorithm) value/weight ratio (descending) If ratios are equal → use stable ordering based on original index But the input is almost sorted except for a few misplaced elements Design the pseudocode for the whole problem (fractional + sorting)

2. You're building a backend system for a major online retailer. Every day, the platform receives millions of customer orders, each with a specific delivery deadline. These orders must be reorganized quickly so that the ones with the earliest deadlines are processed first.

The dataset is too large to load entirely into memory, so the sorting algorithm must be efficient in both time and space. Additionally, the order of entries is often partially shuffled, not completely random, and sometimes nearly sorted.

Apply an algorithm to sort these orders by their deadlines, ensuring that the approach scales well with large inputs and handles partially sorted data gracefully. Make sure that this algorithm doesn't hold any extra space.

3. Go through the following pseudo code, if you find any error then correct it.

void merge(int a[], int p, int q, int r) { int b[5]; //same size of a[] int i, j, k; k = 0; i = p; j = q + 1;
while(i <= q && j <= r) { if(a[i] < a[j]) { b[k++] = a[j++];
} else { b[k++] = a[j++]; } }

```
while(i <= p)
{
   b[k++] = a[i++];
}

while(j <= q)
{
   b[k++] = a[j++];
}

for(i=0; i >= p; i--)
{
   a[j] = b[--k];
```

Instructions: For the problem 2, Compile the code, take the proper screenshots of your input and output code. For problem 1,3 type the answer. Adding cover page and name it as "last 3 digit of ID_Section". & submit it within the time deadline. Make sure your all answers are authentic.