Q3. Implement either the Strategy or Factory Design pattern to sort the below problem.

- The user will enter the student names
- The user will enter the subjects (as many subjects as he/she wants)
- The user will enter the marks for all students for each subject
- You will total the marks for each student
- You will store them in a dynamic list. The user should be able to add as many names, subjects and marks as he/she wants.
- You will prompt the user to select a subject or the total marks on which the list will be sorted.
- You will prompt the user to select an order to sort the list (ascending or descending)
- You will prompt the user to select an algorithm to sort the list. The algorithm options are: Quicksort, Mergesort, Selection Sort, Heapsort.
- Based on the user selection of algorithm, subject or total marks and the order of sorting, you will sort the list with the selected algorithm on the selected subject or total marks and in the order selected by user.
- The class structure must follow either **Strategy or factory design** pattern.

Example:

Input:

User enters the Names = {'Nicole', 'Sam', 'Ryan'}
User enters the Subject = {'English', 'Math', 'Science'}
User enters the Marks = Nicole: {30, 50, 40}, Sam: {50, 70, 50}, Ryan: {40, 40, 30}

Please select a subject or total mark to sort the list (1) English (2) Math (3) Science (4) Total Marks: **4** (Input from user)

Please select an algorithm to sort the list (1) Quicksort (2) Mergesort (3) Selection Sort (4) Heapsort: **1** (Input form user)

Please select an order to sort the list (1) Ascending (2) Descending: **2** (Input from user)

Output:

Rank	Name	English	Math	Science	Total Marks
1	Sam	50	70	50	170
2	Nicole	30	50	40	120
3	Ryan	40	40	30	110

NOTE: You must implement the algorithms (quicksort, mergesort, selection sort & heapsort) to sort the list. DO NOT USE ANY BUILT IN FUNCTION TO SORT THE LIST. AND IT MUST FOLLOW EITHER THE STRATEGY OR FACTORY DESIGN PATTERN.