Sorting

Attention please: don't use STL in C++ and <bits/stdc++.h>, apply algorithm accordingly which we have asked, don't copy each other or don't copy from various website if any two-answer sheet are same then no credit will be given, and one remainder all marks are add for final marks so do it sincerely. First compile successfully then check the answer and run it by given test case if output matches then submit otherwise try to correct it. Don't submit more than one file for each question (no output file required), don't send more than one mail, first do all the question carefully then attach it with your email and send them to notified mail id with your roll number and name.

1. Given an unsorted array, sort it using **insertion sort (in-place)** and print the array. You should take the input from user (i.e. size of the array, array elements).
   Ex: arr[]={4,7,2,9,3} , n=5.
   After sorting, the array should be
   arr[]={2,3,4,7,9}.
2. Given an unsorted array, sort it using **insertion sort (recursion)** and print the array. You should take the input from user (i.e. size of the array, array elements).
   Ex: arr[]={4,7,2,9,3} , n=5.
   After sorting, the array should be
   arr[]={2,3,4,7,9}.
3. Given an unsorted array, sort it using **merge sort (divide and conquer)** and print the array. You should take the input from user (i.e. size of the array, array elements).
   Ex: arr[]={4,7,2,9,3} , n=5.
   After sorting, the array should be
   arr[]={2,3,4,7,9}.
4. Given an unsorted array, sort it using **merge sort (in-place)** and print the array. You should take the input from user (i.e. size of the array, array elements).
   Ex: arr[]={4,7,2,9,3} , n=5.
   After sorting, the array should be
   arr[]={2,3,4,7,9}.
5. Print **the best time complexity, average time complexity, worst time complexity and space complexity** with an example of array of above four questions.
   Ex: you should print the answer in this way for each question
   Question1.
   Insertion sort(in-place):
   Best time: O(*) or Ω(*) or ϴ(*) |**space:** O(*) or Ω(*) or ϴ(*) |**array:** [$,$,$,$,$]
   Worst time: #######| **space complexity:** #######| **array:** #######
   Average time: ######| **space complexity:** #######| **array:** #######
   Where O(*) **– Big O(*)**, Ω(*) – **Big Omega(*)**, ϴ(*) – **Big Theta(*)**, here * may be n, n^2,n^3,..etc. n= size of the array and $ is the element of the array.