In many applications we want to find the k smallest (or largest) elements in an array of n numbers (for example the top 5 recommendations for movies to rent, or the top 3 GPAs at UTEP, and so on). Your task it to implement several solutions to this problem. First, ask the user to enter n and k, with k << n. Then generate a random array A with n elements, and finally print the k smallest elements in the array (if k==1, your method prints the smallest element in A, if k==2 it prints the smallest and second smallest, and so on).

To find the smallest elements, implement the following algorithms

1. Sort A using selection sort; output the first k elements of A.
2. Sort A using mergesort; output the first k elements of A.
3. Run k iterations of the inner loop of bubble sort, starting from the end; output the first k elements of A.
4. Run k iterations of selection sort. In the first iteration you find the smallest element in A and swap it with A[0], in the second iteration you find the smallest element in A[1],…,A[A.length-1], and swap it with A[1], and so on. Then you output the first k elements of A.
5. Have a separate sorted array S containing the k smallest elements found so far. First copy the first k elements of A into S, but store them in ascending order (so S[0] is the smallest element in A[0],…,A[k-1], and S[k-1] is the largest). Then, scan the remaining part of A, and every time you find an element that is smaller than the largest element of S (that being S[k-1]) you insert that element into its right position in S, bumping out S[k-1]. Then you output the elements of S.

Write a report describing your work. For every method, determine the big-O running time in terms of n and k, and run experiments with various values on n and k to determine if their analytical running times agree with what you see in practice. Use graphs or plots to illustrate this.