Question - 1
Sorting

SCORE: **5 points**

Please choose all the correct statements from the following selections:

[✓ Switching to insertion sort for small subarrays will improve the running time of a typical mergesort implementation.]

[✓ Insertion sort uses N^2/4 compares and N^2/4 exchanges to sort a randomly ordered array of length N with distinct keys, on the average.]

A large array appended to a small sorted array is a typical example of a partially sorted array.

[✓ Binary search can be used in an insertion sort algorithm to reduce the number of comparisons]

Question - 2
Shell Sort

SCORE: **5 points**

Regarding Shell sort: given the following list of numbers: [5, 16, 20, 12, 3, 8, 9, 17, 19, 7], which answer illustrates the contents of the list after all swapping is complete for a gap size of 3?

- [3, 7, 5, 8, 9, 12, 19, 16, 20, 17]
- [✓ [5, 3, 8, 7, 16, 19, 9, 17, 20, 12]]
- [5, 16, 20, 3, 8, 12, 9, 17, 20, 7]
- [3, 5, 7, 8, 9, 12, 16, 17, 19, 20]

Question - 3
Sorting Complexity

SCORE: **5 points**

What is the theoretical minimum number of compares that a compare-based sorting must perform when data is randomly shuffled prior to the sort?

- N^2
- N
- N!

Help

[✓ N * logN]

Question - 4

Stable Sort

SCORE: **5 points**

Consider an employee database where people are added based on hire date and then sorted based on last name. If two people in the employee database have the same last name, which sorting method(s) will not change their order relative to each other?

- Selection sort
- [✓ Insertion sort]
- Shell sort
- All of the above

Question - 5

Find the median

SCORE: **30 points**

merge sort median

All of you know the concept of a median. If not, When a distribution is sorted, the middle value(or mean of the middle two values) is known as the median.

```
Example:
When the number of elements is odd :[1,2,4,5,7]
The median = 4


When the number of elements is even :
[1,2,4,5,7,8]
The median = 4.5
```

This program finds the median of an array. The logic for finding the median and mergeSort() part is provided. All you have to do is implement the **merge()** methods. Use the basic merge sort methodology--don't apply any of the optimizations.

Needless to say, the mergeSort() implements the logic for the sort and merge() merges the two arrays.
The array **a[] contains the actual list of elements while tempA[] is an auxiliary array for the merge sort.**

```
Sample Input :
3
8 6 7

The first line is size of the array.
The second line is the value of the integers in
the array, ie. The elements in the array.


Sample output :
```

```
7.0

The output is the median of the array [8,6,7]
(from the example input above)
```

NOTE: you are NOT allowed to use the Arrays.sort() method.