Which of the following statements are true?

|  |  |
| --- | --- |
| 1. | The number of comparisons for a selection sort is represented by the series: *(n - 1) + (n*  *- 2) + ... + 2 + 1* |

|  |  |
| --- | --- |
| 2. | Insertion sort is considered a quadratic sort. |

|  |  |
| --- | --- |
| 3. | With respect to selection sort, the number of comparisons is O(n). |

|  |  |
| --- | --- |
| 4. | The improvement of Shell sort over insertion sort is much more significant for small  arrays. |

|  |  |
| --- | --- |
| 5. | The method *sort(int[] items)*, in class java.util.Arrays, sorts the array item in ascending  order. |

Fill out the blanks:

|  |  |
| --- | --- |
| 6. | A class that implements the Comparable interface must define a(n)  method that determines the natural ordering of its objects. |

|  |  |
| --- | --- |
| 7. | Selection sorts an array by making several passes through the array,  selecting the next smallest item each time and placing it where it belongs in the array. |

|  |  |
| --- | --- |
| 8. | In the best case, selection sort makes O(N) comparisons. |

|  |  |
| --- | --- |
| 9. | With respect to merge sort, additional space usage is O( ). |

|  |  |
| --- | --- |
| 10. | The idea behind sort is to sort many smaller subarrays using  insertion sort before sorting the entire array. |

|  |  |
| --- | --- |
| 11. | sort has O(n3/2) or better performance. |

|  |  |
| --- | --- |
| 12. | Insertion sort is an example of a(n) quadratic sorting algorithm. |

|  |  |
| --- | --- |
| 13. | You can think of the sort as a divide-and-conquer approach to insertion sort. |

|  |  |
| --- | --- |
| 14. | The following represents the sort algorithm.  Set the initial value of *gap* to n / 2. while gap > 0  for each array element from position *gap* to the last element Insert this element where it belongs in its subarray.  if *gap* is 2, set it to 1. else *gap* = *gap*/2.2. |
| A) | Shell |
| B) | Heap |
| C) | Insertion |
| D) | Quick |

|  |  |
| --- | --- |
| 15. | The following is the algorithm.  Access the first item from both sequences. while not finished with either sequence  Compare the current items from the two sequences, copy the smaller current item to the output sequence, and  access the next item from the input sequence whose item was copied Copy any remaining items from the first sequence to the output sequence. Copy any remaining items from the second sequence to the output sequence. |
| A) | Shell sort |
| B) | Selection sort |
| C) | Merge sort |
| D) | Heapsort |

|  |  |
| --- | --- |
| 16. | In merge sort, the total effort to reconstruct the sorted array through merging is . |
| A) | O(1) |
| B) | O(log2n) |
| C) | O(n log n) |
| D) | O(n2) |

|  |  |
| --- | --- |
| 17. | Which of the following sorts is not O(n lg(n))? |
| A) | selection |
| B) | heap |
| C) | merge |

|  |  |
| --- | --- |
| 18. | The best sorting algorithms provide average-case behavior and are considerably  faster for large arrays. |
| A) | O(1) |
| B) | O(n) |
| C) | O(n2) |
| D) | O(n log n) |

|  |  |
| --- | --- |
| 19. | Which of the following generally gives the worst performance? |
| A) | Selection sort |
| B) | Bubble sort |
| C) | Insertion sort |
| D) | Shell sort |

|  |  |
| --- | --- |
| 20. | Shell sort is if successive powers of 2 are used for *gap*. |
| A) | O(log n) |
| B) | O(1) |
| C) | O(log2n) |
| D) | O(n2) |

|  |  |
| --- | --- |
| 21. | The following represents the algorithm for sort.  for each array element from the second (nextPos = 1) to the last  Insert the element at nextPos where it belongs in the array, increasing the length of the sorted subarray by 1 element. |
| A) | merge |
| B) | shell |
| C) | selection |
| D) | insertion |

|  |  |
| --- | --- |
| 22. | The following is the algorithm.  Build a heap by rearranging the elements in an unsorted array. while the heap is not empty  Remove the first item from the heap by swapping it with the last item in the heap and restoring the heap property. |
| A) | Quicksort |
| B) | Bubble sort |
| C) | Merge sort |
| D) | In-Place Heapsort |