1) (5 pts) DSN (Recursive Coding)

Write a ***recursive*** function that returns the sum of all of the even elements in an integer array ***vals***, in between the indexes ***low*** and ***high***, inclusive. For example, for the function call sumEven(vals, 3, 8) with the array vals shown below, the function should return 24 + 8 + 10 = 42, since these three numbers are the only even numbers stored in the array in between index 3 and index 8, inclusive.

index	0	1	2	3	4	5	6	7	8	9
vals[i]	15	13	28	19	24	8	7	99	10	14

```
int sumEven(int vals[], int low, int high) {

    if (low > high) return 0;        // 1 pt base case.

    int res = 0;                     // 2 pts assign variable based
    if (vals[low]%2 == 0)            // on current term.
        res = vals[low];

     return res + sumEven(vals, low+1, high); // 2 pts return and
                                              // recursive call

}
```

2) (10 pts) ALG/DSN (Sorting)

(a) (5 pts) Consider running a Bubble Sort on the array shown below. How many swaps will execute for the duration of the algorithm running on the array shown below? Explain how you got your answer.

97	16	45	63	13	22	7	58	72

Reasoning:

On iteration #1, 97 will swap with all items, for a total of 8 swaps.
On iteration #2, 63 will swap with 13, 22, 7 and 58, for a total of 4 swaps.
On iteration #3, 45 will swap with 13, 22 and 7 for a total of 3 swaps.
On iteration #4, 16 swaps with 13, and 22 will swap with 7 for a total of 2 swaps.
On iteration #5, 16 will swap with 7, for a total of one swap.
On iteration 36, 13 will swap with 7, for a total of one swap.

Adding up, we get 8 + 4 + 3 + 2 + 1 + 1 = 19 swaps.

A perhaps, easier way to solve this is to realize that all swaps are between inverted elements, namely, pairs of items that are out of place in the original array, meaning that in the pair the larger item appears first. Thus, we can count all the inversions:

(97, 16), (97, 45), (97, 63), (97, 13), (97, 22), (97, 7), (97, 58), (97, 72),
(16, 13), (16, 7),
(45, 13), (45, 22), (45, 7),
(63, 13), (63, 22), (63, 7), (63, 58)
(13, 7)
(22, 7)

which is 19 inverted pairs.

Grading: 4 pts for correct answer, 1 pt for reason. If answer is 18 or 20 and reason is valid, take off 1 pt, if answer is 17 or 21 and reason is valid, take off 2 pts.

(b) (5 pts) List the **best case** run time of each of the following sorting algorithms, in terms of n, the number of items being sorted. Assume all items being sorted are distinct.

(i) Insertion Sort O(n) **Grading: 1 pt each**

(ii) Selection Sort $O(n^2)$

(iii) Heap Sort O(nlgn)

(iv) Merge Sort O(nlgn)

(v) Quick Sort O(nlgn)