1. (5 points) Design an algorithm for computing $\lfloor \sqrt{n} \rfloor$ for any positive integer n. Besides assignment and comparison, your algorithm may only use the four basic arithmetical operations.

2. (5 points) Design an algorithm to find all the common elements in two sorted lists of numbers. For example, for the lists $2, 5, 5, 5$ and $2, 2, 3, 5, 5, 7$, the output should be $2, 5, 5$.What is the maximum number of comparisons your algorithm makes if the lengths of the two given lists are m and n, respectively?

3. (5 points) Describe the standard algorithm for finding the binary representation of a positive decimal integer in pseudocode.

4. (5 points) Consider the following algorithm for finding the distance between the two closest elements in an array of numbers. Make as many improvements as you can in this algorithmic solution to the problem. If you need to, you may change the algorithm altogether; if not, improve the implementation given.

```
ALGORITHM MinDistance(A[0..n − 1])
//Input: Array A[0..n − 1] of numbers
//Output: Minimum distance between two of its elements
dmin ← ∞
for i ← 0 to n − 1 do
        for j ← 0 to n − 1 do
                if i ≠ j and |A[i] − A[j ]| < dmin
                        dmin ← |A[i] − A[j ]|
return dmin
```

5. (5 points) Design an algorithm for checking whether two given words are anagrams, i.e., whether one word can be obtained by permuting the letters of the other. For example, the words ***tea*** and ***eat*** are anagrams.

6. (5 points) Consider the algorithm for adding two $n \times n$ matrices. What is its basic operation? How many times is it performed as a function of the matrix order n (input size is order of matrix)? How many times is it performed as a function of the total number of elements in the input matrices (input size is number of elements in both matrices)?

7. (5 points) Consider the algorithm for **multiplying** two $n \times n$ matrices. What is its basic operation? How many times is it performed as a function of the matrix order n? How many times is it performed as a function of the total number of elements in the input matrices?

8. (5 points) Write an algorithm variation of sequential search that scans a list to return the number of occurrences of a given search key in the list. Does its efficiency differ from the efficiency of classic sequential search? Analyze it in all cases possible.

9. (5 points) Suggest how any sorting algorithm can be augmented in a way to make the best-case count of its key comparisons equal to just $n - 1$ (n is a list's size, of course). Do you think it would be a worthwhile addition to any sorting algorithm?

10. (5 points) For each of the following pairs of functions, indicate whether the first function of each of the following pairs has a lower, same, or higher order of growth (to within a constant multiple) than the second function. You should argue your answer and can use any approach you like to do so.
    a. $n(n+1)\,,\ 2000n^2$
    b. $n^2\,,\ n^3$
    c. $\log n\ \ ,\ln n$
    d. $(n-1)!\ ,\ 2^n$

11. (5 points) Compute the following sums.
    a. $1 + 3 + 5 + 7 + \ldots + 999$
    b. $\sum_{i=3}^{n+1} i$
    c. $\sum_{i=0}^{n-1} i(i+1)$
    d. $\sum_{i=1}^{n} \sum_{j=1}^{n} ij$

12. (5 points) Find the order of growth of the following sums. Use the $\Theta(g(n))$ notation with the simplest function $g(n)$ possible.
    a. $\sum_{i=0}^{n-1} (i^2+1)^2$
    b. $\sum_{i=2}^{n-1} \lg i^2$
    c. $\sum_{i=1}^{n} (i+1)2^{i-1}$
    d. $\sum_{i=0}^{n-1} \sum_{j=0}^{i-1} (i+j)$

13. (5 points) Consider the following algorithm.

$S \leftarrow 0$
$for\ i \leftarrow 1\ to\ n\ do$
    $s \leftarrow s + i * i$
$return\ S$

a. What does this algorithm compute?
b. What is its basic operation?
c. How many times is the basic operation executed?
d. What is the efficiency class of this algorithm?

14. (5 points) Suggest an improvement or a better algorithm altogether for the previous question and indicate its efficiency class. If you cannot do it, try to prove that, in fact, it cannot be done.

15. (5 points) Solve the following recurrence relations.

a. $x(n) = 3x(n-1)\ for\ n > 1, x(1) = 4$
b. $x(n) = x(n-1) + n\ for\ n > 0, x(0) = 0$
c. $x(n) = x(n/2) + n\ for\ n > 1, x(1) = 1\ (solve\ for\ n = 2^k)$
d. $x(n) = x(n/3) + 1\ for\ n > 1, x(1) = 1\ (solve\ for\ n = 3^k)$

16. (5 points) Consider the following recursive algorithm.

ALGORITHM $Q(n)$
$//Input: A\ positive\ integer\ n$
if $n = 1$ **return** 1
else return $Q(n-1) + 2 * n - 1$

a. Set up a recurrence relation for this function's values and solve it to determine what this algorithm computes.
b. Set up a recurrence relation for the number of multiplications made by this algorithm and solve it.
c. Set up a recurrence relation for the number of additions/subtractions made by this algorithm and solve it.