1. For the following algorithm write a recurrence relation for its time complexity and give its asymptotic behavior using Θ notation:

```
alg(n)
for i=1 to n
   alg(n-1)
Θ(1)
```

2. Order the following functions with respect to growth rate (you can use O, Ω, o, ω, and Θ
   (a) n^3 and $\lg^2 n$
   (b) n^2 and $2^{\sqrt{2 \lg n}}$
   (c) $\ln n$ and $\lg n$
   (d) $(3/2)^n$ and 2^n
3. Solve the following recurrence relations:
   (a) $T(n) = T(n-2) + n^2$
   (b) $T(n) = 2T(n/2) + \frac{n}{\lg n}$
   (c) $T(n) = T(n-2) + \frac{1}{\lg n}$
4. Modified partition.
   (a) Write a modified version of the partition algorithm that splits the list in three parts instead of two, using the first and last elements of the list as pivots. Your modified partition algorithm must work in linear time.
   (b) Write a modified version of quicksort that uses your modified partition algorithm.
   (c) Write a recurrence relation for the best case time complexity of your modified quicksort algorithm.
   (d) Solve the recurrence relation for your modified quicksort algorithm
5. Given the following array, show a tree containing all the pivots used by the quicksort (with partition) algorithm. The elements must be sorted alphabetically

   W Q P V R Z Y U T B
6. Given a sorted array A[1...n]
   (a) Write a divide and conquer algorithm that determines in $O(\lg n)$ if $\exists i \in 1, .., n(A[i] = i)$ (the integers in the array may be negative!)
   (b) Explain why your algorithm is correct (no need for a formal proof, just write an idea)
   (c) Write a recurrence relation that describes the running time of your algorithm.
   (d) solve your recurrence relation from the previous part.

7. Suppose that you have a linear time "partition" algorithm that always divides a list in two parts such that:

   - All the elements in the first part are smaller than all the elements in the second one, and
   - The largest part always has at most 60% of the elements of the original array.

   (a) Write a **RECURSIVE** algorithm that finds the k-th smallest element and that **USES** the "partition" algorithm given above.

   (b) Write a recurrence relation for the worst-case running time of your algorithm.

   (c) Does this algorithm work in worst case linear time?