1- (Find a Peak)
   Given an array A of integers. Find a peak element in A. An array element is a peak if it is
   NOT smaller than its neighbors. For corner elements, we need to consider only one
   neighbor.
   a) Implement two different solutions for the peak finding algorithm and show that a better
   algorithm impacts on the running time of the algorithm.
   b) Consider a matrix A as an array and suppose that a[i,j ] is a peak if and only if a[i-1]<a[i]>=a[i+1] and a[j-1]<a[j]>=a[j+1]. i, j indicate the row and column of a[i,j].
   Design an algorithm to find a peak in A. try the best way possible to minimize the time
   complexity of your algorithm.
   c) Implement your algorithm. And compare it with a worse algorithm in terms of time
   complexity.

2- Given two functions $f = \Omega(\log n), g = O(n)$ consider the following statements. For each
   statement prove if it is true or false.
   a) $f(n) \in \Omega(\log(g(n)))$
   b) $f(n) \in \Theta(\log(g(n)))$
   c) $f(n) \in O(2^{g(n)})$
   d) $f(n) + g(n) \in \Omega(\log n)$

3- Discuss if the following statements are true:
   a) $(\log_2 n)! \in \Omega(n!)$
   b) $nsinm \in \Omega(n)$
   c) $a^{\log n} \in O(\log n^{\log n})$
   d) $(\log_2 n)! \in \Omega(n!)$
   e) $(1 + \varepsilon)^n \in O\left(\frac{n^2}{\log n}\right)$
   f) $e^{c\sqrt{n}} \in O\left(e^{\sqrt{n}}\right), c>0$
   g) $2^{f(n)} \in O\left(2^{g(n)}\right), f(n) \in O(g(n))$
   h) $\log_2 g(n) \in O(\log(f(n))), f(n) \in O(g(n))$ and $\log f(n) \geq 1$
   i) $\log\log^* n \in o(\log * \log n)$
4- Solve the following recursive functions:

a) \( T(n) = \begin{cases} 
   a & n = 0 \\
   b & n = 1 \\
   1 + T(n-1) & n > 1 \\
   \frac{T(n-2)}{4-T(n-1)} & n = 1 \\
\end{cases} \)

b) \( T(n) = \begin{cases} 
   1 & n > 1 \\
   \frac{1}{4-T(n-1)} & n = 1 \\
\end{cases} \)

c) \( T(n) + nT(n-1) = 2n! \), \( T(0) = 1 \)

d) \( T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\log n} \)

e) \( T(n) = T\left(\sqrt{n}\right) + T\left(n - \sqrt{n}\right) + \theta(n) \)

f) \( T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{3}\right) + n \)

g) \( T(n) = 4T\left(\sqrt[n]{n}\right) + \log^2 n \)

h) \( T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{n}{\sqrt[n]{n}}\right) + n\sqrt{\log n} \)

i) \( T(n) = \sum_{i=1}^{n-1} T(i) + 3n, \quad T(1) = 1 \)

ej) \( T(n) = T\left(\frac{2n}{3}\right) + \log^2 n \)

5- Consider the Hanoi Tower problem discussed in the class. Add an extra constraint to the problem: a disk cannot be moved straight from column i to j.

a) Write a pseudo-code for this problem, call it \( H(n) \). \( n \) indicates the number of discs.

b) Write a recursive function that represents the number of movements it needs to move \( n \) discs. Call this function \( F(n) \).

c) Solve the above-mentioned (6b) recursive function.

6- (World Championships)

Consider two teams: A and B. These two teams should play with each other until one of them wins \( n \) times. The winner can go to the next level. Suppose team A wins a game with a probability equals to \( p \), and team B wins with a probability equals to \( q \). Also suppose that one team should win in each game. We want to know with what probability team A will go to the next level.

a) Write a recursive function that indicates the probability that team A may go to the next level.

b) Find the number of ADD instruction in computing \( P(n,m) \). \( P(i,j) \) illustrates the probability that A wins given the condition that team A already has won \( i \) iterations and team B has won \( j \) iterations.

c) Compute the number of multiplications.

d) Write a recursive function to compute the number of calls to “return”, and solve this recursive function.
7- Given an array A that contains n integers with an arbitrary permutations, find the best algorithm possible in terms of time complexity that finds out if the following statement is true for any 4 numbers in A?

8- Analysis the time-complexity of the following loops:
   a) 
   \[
   k \leftarrow 0 \\
   \text{for} \ (i = 1, i \leq n, i++) \\
   \quad \text{for} \ (j = 1, j \leq \frac{n}{i}, j++) \\
   \quad \quad k \leftarrow k + 1 \\
   \]
   b) 
   \[
   x = 0 \\
   \text{for} \ (i = 1, i \leq n, i = i \times 2) \\
   \quad \text{for} \ (j = 1, j \leq n, j = j \times 4) \\
   \quad \quad \text{for} \ (k = 1, k \leq j, k++) \\
   \quad \quad \quad x++ \\
   \]
   c) 
   \[
   x = 0 \\
   \text{for} \ (i = 2, i \leq n, i = i++) \\
   \quad \text{for} \ (j = 1, j \leq n, j = j \times i) \\
   \quad \quad x++ \\
   \]

9- Write a recursive function for the selection sort presented in the class, and analysis the time complexity of that recursive function.

10- Write a recursive function for the insertion sort presented in the class, and analysis the time complexity of that recursive function.
11- Given an array A of n integers:
   a) Implement a program that finds out if there are less than or equal to $\sqrt{n}$ peaks in
      the array (The peak definition is defined in the class, and also the first exercise.),
      then uses insertion-sort to sort A.
   b) If there are $\sqrt{n} < k \leq n^{\frac{3}{4}}$ peaks in the array then choose $|k|$ arrays out of them
      and merge all of the arrays into one sorted array.
   c) For $n^{\frac{3}{4}} < k$. Choose the peaks and use insertion sort to sort the peaks, then move the
      peaks to the right side of A. Do this again for the remained items of A. and find
      peaks and sort them and place them in their correct position, until the size of the
      array in the last iteration gets lower than log n. Use insertion sort and sort the logn
      items.

12- Array A[1..n], that $n = 2^k$, contains all k-bit sequences (k-bit-string) except for one that
    is called x. We do not know about the x sequence. The only operation we can do is “to read
    the jth bit of the A[i] sequence” which can be done in constant time. Find a recursive
    function that can find x in linear time. (Use Radix Sort)

13- Soccer Game
    There are $n = 2^k$ teams, and they need to play $\frac{n}{2}$ games. The same procedure goes on
    between both winners and losers from the previous step in the next level. That is in each
    step, each winner will play with another winner from the previous step, and every loser
    should play with another loser. This procedure goes on until there is only one team in a
    step, so there will be no game. (Each game must have a loser and a winner). How many
    games are there in the whole competition?

14- Consider a Binary Counter. Suppose it costs i$ for changing the ith bit. Prove that the
    amortize of cost of each increase operation is O(1).

15- Given a machine that can compute the kth smallest item of an Array A in $O(\sqrt{n})$ time. Find
    a recursive function that can sort A in linear time corresponding to n which is the length of
    A.