1. Suppose we run quicksort, as presented in the lectures and in the class notes, to sort the following array:

\[ [62, 98, 33, 21, 60, 96, 75, 46] \]

(a) Show the contents of the array after the split step at the top level is complete, immediately before the first recursive call.

(b) Draw the binary tree of lists for this invocation of quicksort, as described in the lectures and the class notes in the analysis of quicksort (slide 2-37).

2. If \( S \) be a set of integers and \( x \) is an integer, the predecessor of \( x \) in \( S \) is defined to be the largest element of \( S \) that is \( \leq x \) if such an integer exists, and \( -\infty \) otherwise.

Consider the following problem:

We are given a set \( S \) of integers, represented as a sorted list. We are also given a list \( Q \) of query values, which is also a sorted list. We want to output a list of pairs, where each query value is paired with its predecessor in \( S \).

As an example, if \( S = \{2, 3, 5, 7, 11\} \) and \( Q = [1, 3, 8, 9, 20] \), the output would be \([(1, -\infty), (3, 3), (8, 7), (9, 7), (20, 11)]\).

Describe an algorithm that solves this problem in \( O(n + k) \) worst-case time, where \( n \) is the number of items in the list \( Q \) and \( k \) is the number of elements in the set \( S \).