Spring 2019 CS372 Assignment #3.

Due: on Thursday, February 21, at the beginning of the lecture.

You may do this assignment in groups of 2 or individually.

1. Consider the following recursive algorithm:

```
Parallel-Product(A[1..n])

if n = 1 then return
for i := 1 to n/2 do
     A[i] = A[i]*A[i+n/2]
Parallel-Product(A[1..n/2])
```

Let T(n) denote the running time of `Parallel-Product` on the input of size n. Derive a recurrence relation for T(n).

2. Recall pseudocode for Quicksort and Partition:

```
Quicksort(A; p; r )
if p < r
     q = Partition(A; p; r )
     Quicksort(A; p; q - 1)
     Quicksort(A; q + 1; r )

Partition(A; p; r )
x = A[r ]   // x is selected as a pivot element
i = p - 1
for j = p to r - 1
     if A[j ] ≤ x
          i = i + 1
          exchange A[i ] with A[j ]
exchange A[i + 1] with A[r ]
return i + 1
```

2.1 Here is an array which has just been partitioned by the first step of Quicksort:

`3, 0, 4, 2, 5, 8, 7, 6, 9`

Which of these elements could have been the pivot? (if there are more than one possibility, list them all)

2.2. What value of q does Partition return when all elements in the array A[p…r] have the same value?

2.3. What is the running time of Quicksort when all elements of array A have the same value? Justify your answer.

2.4. Suppose that the partitioning algorithm always produces a 2-to-7 proportional split. That is, the size of one obtained subproblem divided by the size of the other subproblem is 2/7. Write the recurrence on the running time of Quicksort in this case.

3. Let A and B be two sequences of n integers each. Give a pseudo-code of an O(n log n) algorithm for printing all integers that A and B have in common. For instance, if A is 2, 7, 4, 9, 10, 5, and B is 20, 5, 3, 6, 8, 7, then the algorithm should output 7 and 5. (The order in which the numbers are printed is not important). If a number appears more than once in A and B, then it should be printed as many times as there are common occurrences of it. For instance, if A is 1011 and B is 2113, then the algorithm should output 1 two times. Explain why the running time of your algorithm is O(n log n).

4. Exercise 2.17.

5. Exercise 2.19. Hint for 2.19a): Let T(i) be the time to merge arrays 1 to i. T(i) consists of the time taken to merge arrays 1 to i-1 and the time taken to merge the resulting array of size (i-1)n with array i. Write a recurrence for T(i) and solve it. Hint for 2.19b): Divide the arrays into two sets, each of k/2 arrays.

6. Exercise 2.20.