CSC 421/Applied Algorithms and Structures
Problem set 1

Simulate the MERGE *method*

Input	A = [16,18,28,48,56,64,70,72,4,8,17,25,34,38,68,76]
Divide	A = [16,18,28,48,56,64,70,72 | 4,8,17,25,34,38,68,76]
Recursive Left	A = [16,18,28,48,56,64,70,72 | 4,8,17,25,34,38,68,76]
Recursive Right	A = [16,18,28,48,56,64,70,72 | 4,8,17,25,34,38,68,76]
Merge	A= [4, 8, 16, 17, 18, 25, 28, 34, 38, 48, 56, 64, 68, 70, 72, 76]

Simulate the PARTITION *method*

Input	A = [16,18,28,48,56,64,70,72,4,8,17,25,34,38,68,76]
Choose a pivot	A = [16,18,28,48,56,64,70,72,4,8,17,25,**34**,38,68,76]
Partition	A=[16, 18, 28, 4, 8, 17, 25, **34**, 48, 56, 64, 70, 72, 38, 68, 76]
Recursive left	A = [4, 8, 16, 17, 18, 25, 28, **34**, 48, 56, 64, 70, 72, 38, 68, 76]
Recursive right	A = [4, 8, 16, 17, 18, 25, 28, **34**, 38, 48, 56, 64, 68, 70, 72, 76]

Exercise 24 (a, b, and c) in the Chapter 1 exercises (8 points)

For (a), write down the number of multiplications in terms of n. For (b), write down the number of bits in terms of n.

24. Consider the following classical recursive algorithm for computing the factorial $n!$ of a non-negative integer n:

```
FACTORIAL(n):
  if n = 0
      return 1
  else
      return n · FACTORIAL(n − 1)
```

(a) How many multiplications does this algorithm perform?

(b) How many bits are required to write $n!$ in binary? Express your answer in the form $\Theta(f(n))$, for some familiar function $f(n)$. *[Hint:* $(n/2)^{n/2} < n! < n^n$*.]*

(c) Your answer to (b) should convince you that the number of multiplications is *not* a good estimate of the actual running time of FACTORIAL. We can multiply any k-digit number and any l-digit number in $O(k \cdot l)$ time using either the lattice algorithm or duplation and mediation. What is the running time of FACTORIAL if we use this multiplication algorithm as a subroutine?

A) the algorithm recursively does multiplication for let's say T(n) times. The algorithm performs multiplication once every recursion call is made.

T(n) = 1 + T(n-1)

T(n) = 1 + 1 + T(n-2) = 2 + T(n-2)

T(n) = 1 + 1+ 1 + 1 +….+ T(0) = (n-1) + T(0)

We know that base case has only one step, so T(0) = 1
So we have :

T(n) = n - 1 + 1 = n

Therefore, the algorithm performs **n** multiplications.

B) Let's say you want to represent the number 42 in binary.
That is 2^x that is approximately equal to 42. In order to find the value of exponent x, we use logarithm base 2.

Following the steps below :

- 2^5 (32) is less than 42, but 2^6 (64) is greater than 42.
- Therefore, 42 falls between 2^(6-1) and 2^6.
- Taking the base-2 logarithm of 42 gives us approximately 5.39.
- Since we need the smallest number of bits that can represent 42,
we take the ceiling of 5.3, which is 6.

Utilising the hint given below to find the number of bits required to represent n! in binary:

$$\left(\frac{n}{2}\right)^{\frac{n}{2}} < n! < n^n$$

Apply logarithm base 2 on all functions

$$\log_2 \left(\frac{n}{2}\right)^{\frac{n}{2}} < \log_2 (n!) < \log_2 (n^n)$$

we know that $\log (a^b) = b \log a$

$$\therefore \frac{n}{2} \log_2 \left(\frac{n}{2}\right) < \log_2 (n!) < n \times \log_2 (n)$$

Therefore, the number of bits required to represent n! in binary: (n x log2(n))

$$\boxed{n \times \log_2 (n)}$$

C) Both lattice and Pesanant multiplication algorithm requires O(kl) time to multiply an k-digit number by an l-digit number;
In the first problem we concluded that factorial does multiplication n times.
So we have,

$$T(n) = T(n-1) + O(\log n * n \log n)$$

Here n is the number of bits required by n
And n*log(n) is the number of bits required by n!

$$T(n) = T(n-1) + O(n \log^2 n)$$

Substitute the value of T(n-1) :

$$T(n) = O(n^2 \log^2 n)$$