1) Devise formulas to calculate `my_first_i` and `my_last_i` in the following code snippet from the lecture.

   Assume that `p` (number of cores) and `n` (number of values) are available to all cores, and each core has a value `my_id`, which is a unique identifier between 0 and `n`-1. You can also assume `n` is divisible by `p`

```
my_sum = 0;
my_first_i = ...;
my_last_i = ...;

for (my_i=my_first_i; my_i<my_last_i; my_i++)
{
  my_x = test_even(a[my_i]);
  my_sum += my_x;
}
```

   Your answer needs to depend on my_id, n and p

2) We saw that in the case that p is 8, Core 0 had 7 receives and adds for algorithm1 (Core 0 does all the additions for global sum) and 3 for algorithm2 (tree-structured addition).

   a) Complete the following table with the numbers of receives and additions as p varies

p	2	4	8	16	32	64	128	256	512	1024
Algo1			7							
Algo2			3							

   b) Derive formulas (based on `p`) for the number of receives and additions that core 0 carries out for algorithm1 and algorithm2

3) Go back to question 1. What if n is not divisible by p? Write code instructions to determine my_first_i and my_last_i in this case.

4) **(Extra Credit)** Write a pseudo-code algorithm for the tree-structured algorithm **(Algorithm 2)**

You can assume that the number of cores `p` is a power of 2.