Задание 1

> Доказать, что
>
> $$\sum_{i=0}^{\log n} \log\left(\frac{n}{2^i}\right) = \Theta(\log^2 n)$$

$$\sum_{i=0}^{\log n} \log\left(\frac{n}{2^i}\right) = \sum_{i=0}^{\log n} \log(n) - i = (\log(n)+1)\cdot\log(n) - \sum_{i=0}^{\log n} i =$$

$$= (\log(n)+1)\cdot\log(n) - \log(n)\cdot\frac{\log(n)+1}{2}$$

$$= \frac{(\log(n)+1)\cdot\log(n)}{2} = \frac{1}{2}\log^2(n) + \frac{1}{2}\log(n)$$

Для $n > 4$ верно:

$$\frac{1}{2}\log^2(n) \le \frac{1}{2}\log^2(n) + \frac{1}{2}\log(n) \le 2\log^2(n)$$

$$\frac{1}{2}\log^2(n) + \frac{1}{2}\log(n) \in \Theta(\log^2(n))$$

Задание 2

> Написать код алгоритма scan в fork-join модели с $O(n)$ work и $O(\log n)$ span.

```go
func Scan(a []int, f func(int, int) int) []int {
  n := len(a)
  sum := make([]int, 2*n-1)
  result := make([]int, n)

  ScanUp(a, sum, 0, 0, n, f)
  ScanDown(a, sum, result, 0, 0, n, 0, f)

  return result
}

func ScanUp(
  a, sum []int,
  node, l, r int,
  f func(int, int) int,
) int {
  if r-l == 1 {
    sum[node] = a[l]
    return sum[node]
  }
  m := (l + r) / 2
  var left, right int
  Fork2Join(
    func() { left = ScanUp(a, sum, 2*node+1, l, m, f) },
    func() { right = ScanUp(a, sum, 2*node+2, m, r, f) },
  )
```

```
27    sum[node] = f(left, right)
28    return sum[node]
29  }
30
31  func ScanDown(
32    a, sum,
33    res []int,
34    node, l, r, fromLeft int,
35    f func(int, int) int,
36  ) {
37    if r-l == 1 {
38      res[l] = fromLeft + a[l]
39      return
40    }
41
42    m := (l + r) / 2
43
44    leftSum := sum[2*node+1]
45
46    Fork2Join(
47      func() { ScanDown(a, sum, res, 2*node+1, l, m, fromLeft, f) },
48      func() { ScanDown(a, sum, res, 2*node+2, m, r, fromLeft+leftSum, f) },
49    )
50  }
```

Задание 3

> Вам нужно написать функцию $\text{eps}(a, S)$, которая получает положительных чисел a и возвращает минимальное j, такое что сумма элементов $a[1] + ... + a[j] \geq S$, за $O(j)$ work и $O(\log^2 j)$ span. Предподсчётом пользоваться нельзя.

Идея алгоритма:

1. делаем Reduce на блоках $\left(2^k - 1, 2^{k+1}\right)$, где $k \in [0, \lfloor \log j \rfloor]$

$$\text{work} = 2^{k+1} = O(j)$$

$$\text{span} = \sum_{0}^{\lceil \log j \rceil} \log\left(\frac{j}{2^i}\right) = O(\log^2 j)$$

1. дальше на отрезке $\left(2^k - 1, 2^{k+1}\right)$, где $k = \lfloor \log j \rfloor$ делаем Scan + бинпоиск j по префексным суммам

$$\text{work} = O(j)$$

$$\text{span} = O(\log(j))$$

```
1  func EPS(a []int, S int) int {
2    prev_k := 0
3    next_k := 1
4    sum := 0
5
6    for {
```

```
    partial_sum := Reduce(prev_k, next_k, a)
    if partial_sum+sum >= S {
      break
    }
    sum += partial_sum
    prev_k = next_k + 1
    next_k = min(next_k * 2, len(a))
  }

  prefSums := Scan(a[prev_k:next_k])

  prefSumIndex := BinarySearch(
    prefSums, S,
    func(idx int) bool { return prefSums[idx]+sum >= S },
  )

  return prefSumIndex + prev_k
}
```

Задание 4

> Опишите алгоритм нахождения всех простых до N за $O(N \log\log N)$ work и $O(\text{polylog } N)$ span. (Желательно за $O(\log N \cdot \log\log N)$ span). Тут можно пользоваться parallel_for, map, scan и filter. Псевдокод даже необязательно.

Идея в том чтобы вычислять решето Эратосфена и с помощью `parallel_for` вычеркивать кратные числа, для этого создается булевый массив в котором помечаются числа которые не могут быть простыми:

1. Если `N==2` возвращаем 2
2. Ищем рекурсивно простые до $\sqrt{N}$
3. Вычеркиваем с `parallel_for` все простые от $\sqrt{N}$ до N
4. Фильтруем и возвращаем простые

- Span

$$\text{глубина рекурсии } N^{\frac{1}{\text{Depth}}} = 2 \Rightarrow \text{Depth} = \log\log N$$

$$\text{пометка составных и фильтрация простых требует } O(\log N) \Rightarrow \text{span} = O(\log N \cdot \log\log N)$$

- Work

$$\text{work} = O(N \log\log N) \text{ - обычная сложность решета эратосфена}$$