Задание 1

> Значение X хранится где-то в памяти EREW PRAM. Покажите, как скопировать X в каждую ячейку массива длины p в EREW PRAM с p процессами. Определите, за сколько можно сделать тоже самое в CREW и CRCW PRAM.

1. EREW PRAM:
   1. первый процессор читает X и записывает первое значение массива $A[1]$
   2. превый и второй процессор читают независимо X и $A[1]$ и записывают $A[2]$ $A[3]$
   3. аналогично происходит удвоение масива на следующих этапах

$$\text{time} = 2 \cdot \lceil \log_2 p \rceil$$

2. CREW PRAM: i-процессор конкурентно читает X своей первой опирацией и независимо записывает $A[i]$ на второй

$$\text{time} = 2$$

3. CRCW PRAP: i-процессор конкурентно читает X своей первой опирацией и независимо записывает $A[i]$ на второй

$$\text{time} = 2$$

Задание 2

> У вас есть табличка c[n][m] с n строками и m столбцами. Представьте, что вы - черепашка, которая начинает на верхней строчке длины m и пытается ползти вниз. За один шаг можно спуститься на следующую строчку, а также сдвинуться на −1, 0 и 1 по горизонтали (есть три хода: вниз-влево, вниз, вниз-вправо). Оказавшись в ячейке, черепашка получает число монеток столько, сколько написано в ячейке. Задача узнать - какое максимальное число монеток можно получить, если доползти до самого низа. Напишите EREW алгоритм с асимптотикой O(n + log m) при m процессах.

```
# для процесса id

# копируем табилцу для избежания конкурентного чтения O(n)
for i in 1..n {
  A[i][id] = C[i][id]
  B[i][id] = C[i][id]
}

# обход таблички O(n)
 for i in 1..n {
   # для процессоров id==0 и id==n-1, код должен учитывать границы
   # но думаю и без этого суть алгоритма понятна
   if mod(id, 3) == 0 {
     x = A[i-1][id-1]
   }
   if mod(id, 3) == 1 {
     x = B[i-1][id-1]
   }
   if mod(id, 3) == 2 {
     x = C[i-1][id-1]
   }

```

```
  if mod(id, 3) == 0 {
    z = A[i-1][id]
  }
  if mod(id, 3) == 1 {
    z = B[i-1][id]
  }
  if mod(id, 3) == 2 {
    z = C[i-1][id]
  }

  if mod(id, 3) == 0 {
    z = A[i-1][id+1]
  }
  if mod(id, 3) == 1 {
    z = B[i-1][id+1]
  }
  if mod(id, 3) == 2 {
    z = C[i-1][id+1]
  }

  m = max(x, y, z) + C[i][id]

  A[i][id] = m
  B[i][id] = m
  C[i][id] = m
}

# максимум O(log(m))
D[id] = C[n-1][id]
for i in 1..log(m) {
  if id <= n/(2^i) {
    x = D[2*id+1]
    y = D[2*id]
    D[id] = max(x, y)
  }
}
```

Задание 3

> Докажите, что level-by-level шедулер из теоремы Брента работает не хуже, чем в 2 раза от оптимального.

$$T_{\text{opt}} \geq \left\lceil \frac{W}{P} \right\rceil > \frac{W}{P}$$

1. так как невозможно выполнить работу быстрее чем полностью загрузив каждый поток задачами

$$T_{\text{opt}} \geq S$$

2. так как невозможно выполнить всю работу быстрее чем наиболее длинный путь в графе из последовательных задач

$$T_b < \frac{W}{P} + S < 2 \cdot T_{\text{opt}}$$