Tournament Tree Mutex [tree] - 5 баллов

Реализуйте алгоритм взаимного исключения для n потоков на основе турнирного дерева и алгоритма Петерсона.

Конструктор:
tree_mutex::tree_mutex(std::size_t num_threads)

Методы:
tree_mutex::lock(std::size_t thread_index)
tree_mutex::unlock(std::size_t thread_index)

Параметр thread_index принимает значения от 0 до num_threads-1.

Пример использования:

```
tree_mutex mtx(num_threads);

mtx.lock(0);
// critical section
mtx.unlock(0);
```

Для хранения бинарного дерева используйте его линейную развертку в массив, как в реализации двоичной кучи/пирамиды.

Помните, что алгоритм Петерсона корректно работает только для потоков с индексами 0 и 1, в то время как потоки tree_mutex имеют индексы от 0 до n-1.

Можно считать, что мьютекс будет использоваться корректно, т.е. mtx.unlock(t) будет вызван только если поток с индексом t владел мьютексом.

В секции ожидания в мьютексе Петерсона используйте вызов std::this_thread::yield(), который в случае невозможности захватить мьютекс будет переключать текущее ядро на выполнение другого потока:

```
   void lock(int t) {
        want[t].store(true);
        victim.store(t);
        while (want[1 - t].load() && victim.load() == t) {
            std::this_thread::yield();
        }
    }
```

В противном случае при исполнении на одном ядре поток, который завис в wait-секции, будет целый квант времени греть воздух и не давать другим потокам исполняться.