Семинар #1: Потоки. Домашнее задание.

Задача 1. Поиск максимума

В файле `code/00problem_paralel_max.cpp` написана функция:

```
uint64_t getMax(const std::vector<uint64_t>& v)
```

которая принимает на вход вектор чисел и возвращает максимальный элемент в этом векторе.
Напишите функцию:

```
uint64_t getMax(int n, const std::vector<uint64_t>& v)
```

которая будет делать то же самое, но параллельно, используя `n` потоков.
Протестируйте функцию, замерив скорость работы однопоточной и многопоточной версии.

Задача 2. Поиск максимума на диапазоне

В файле `code/01problem_paralel_max_iterators.cpp` написана шаблонная функция:

```
template <typename RandIt>
RandIt getMax(RandIt start, RandIt finish)
```

которая принимает на вход два random-access итератора и находит максимальный элемент на диапазоне, задаваемом этими итераторами. Функция возвращает итератор на максимальный элемент.
Напишите шаблонную функцию:

```
template <typename RandIt>
RandIt getMax(int n, RandIt start, RandIt finish)
```

которая будет делать то же самое, но параллельно, используя `n` потоков.
Протестируйте функцию, замерив скорость работы однопоточной и многопоточной версии.

Задача 3. Параллельный `sort`

Напишите функцию:

```
template <typename RandIt, typename Comparator>
void parallelSort(int n, RandIt start, RandIt finish, Comparator comp)
```

которая будет сортировать диапазон, задаваемый итераторами `start` и `finish`, используя компаратор `comp`. Алгоритм должен работать в `n` потоков. Например, следующий вызов программы:

```
parallelSort(4, v.begin(), v.end(), [](int a, int b) {return a > b;});
```

должен сортировать вектор целых чисел `v` по убыванию, используя 4 потока.
При реализации этой функции можно использовать однопоточную версию функции `std::sort`.
Протестируйте функцию, замерив скорость работы однопоточной и многопоточной версии.