# Теория:

1. **Параллелизм и конкурентность.**
   Что такое параллелизм и что такое конкурентность? Что такое процесс и что такое поток? Организация параллелизма с использование процессов и с использованием потоков. В чём преемущества и недостатки этих подходов. Зачем нужен параллелизм?

2. **Потоки. Класс `std::thread`**
   Что такое поток? Создание нового потока в языке `C++` с использованием объекта класса `std::thread`. Методы `join` и `detach`. Что произойдёт если выбросится исключение (в новом потоке, или в потоке, который создаёт новый поток)? Передача аргументов в функцию потока. Возврат данных из нового потока. Передача владения потоком. Создание произвольного количества потоков. Идентификация потоков.

3. **Состояние гонки и мьютексы.**
   Что такое разделяемые данные? Что такое состояние гонки (race condition)? Проблематичные и безобидные состояния гонки. Что такое гонка данных (data race) и к чему она приводит? Защита разделяемых данных с помощью мьютекса. Класс `std::mutex`. Методы `lock`, `unlock` и `try_lock`. В чём недостатки класса `std::mutex`? Класс `std::lock_guard`. В чём преимущество `std::lock_guard` перед `std::mutex`? Класс `std::unique_lock`. В чём преимущества и недостатки `std::unique_lock` перед `std::lock_guard`? Взаимоблокировка (deadlock). Решение проблемы взаимоблокировки с помощью стандартной функции `std::lock`.

4. **Механизмы синхронизации.**
   Условные переменные. Класс `std::condition_variable` и как им пользоваться? Методы `wait`, `notify_one` и `notify_all`. Ложные пробуждения (spurious wake). Запуск асинхронной задачи с помощью функции `std::async`. Возврат значения из фоновой задачи с помощью объекта класса `std::future`. Класс задачи – `std::packaged_task`. Зачем могут понадобиться объекты класса `std::packaged_task`? Методы класса `std::packaged_task`: `get_future`, `operator()`. Передача объекта класса `std::packaged_task` в другие функции и потоки. Класс `std::promise`. Методы класса `std::promise`: `get_future`, `set_value` и `set_exception`.

5. **Потокобезопасные стек и очередь с блокировками.**
   Что такое потокобезопасная структура данных? Являются ли стандартные контейнеры `STL` потокобезопасными? Стандартный класс `std::stack` и его методы `push`, `top` и `pop`. Почему в стандартной библиотеке языка `C++` стек реализован так, как он реализован? Что такое потокобезопасная структура данных с блокировками? Написание своего потокобезопасного стека с блокировками. Реализация методов `push` и `pop` такого стека. Потокобезопасная очередь с блокировками. Реализация методов `push`, `try_pop` и `wait_and_pop` такой очереди.

6. **Атомарные типы и операции над ними.**
   Атомарные переменные. В чём отличие атомарных переменных от обычных переменных? Класс `atomic_flag` и его методы `clear` и `test_and_set`. Атомарные типы `atomic<T>` и методы `load`, `store` и `compare_exchange`. Упорядочение доступа к памяти. Упорядочения `memory_order_seq_cst`, `memory_order_acquire`, `memory_order_release` и `memory_order_relaxed`. Реализация спинлока (простейшего мьютекса) на основе атомарной переменной.