Lock-Free Stack

Реализуйте простейший лок-фри стек.
Стек должен поддерживать всего две операции – `void Push(T element)` и `bool Pop(T& element)`.
Как вам должно быть известно, освобождение памяти в лок-фри структурах данных – нетривиальная задача. В данном случае вам предлагается реализовать простейший подход к управлению памяти: складывайте все извлеченные узлы в мусорный список и физически освобождайте их в деструкторе стека.
В вашей реализации не должно быть гонок и утечек памяти.

Lock-Free Queue

Реализуйте классическую лок-фри очередь имени Майкла-Скотта.

Очередь должна поддерживать две операции – `void Enqueue(T element)` и `bool Dequeue(T& element)`.
Как вам должно быть известно, освобождение памяти в лок-фри структурах данных – нетривиальная задача.
Предлагается реализовать механизм, основанный на моментах покоя (quiescent periods):

- Извлеченные узлы накапливаются в мусорном списке, который эпизодически очищается операцией `Dequeue` в тот момент, когда она поняла, что конкурирующих операций нет (наступил момент покоя), а значит узлы в мусорном списке читать никто не может.
- Для отслеживания моментов покоя нужно использовать

счетчик текущих операций.

- Не нужно поддерживать отдельный мусорный список, в силу свойств очереди все извлеченные узлы уже провязаны в список.

Для отладки непосредственно очереди рекомендуется сначала реализовать простейшую схему с удалением всего мусора в деструкторе, и только после этого переходить к реализации более сложного метода освобождения памяти.

В вашей реализации не должно быть гонок и утечек памяти.

Ваша реализация будет параметризоваться типом атомика: по умолчанию это будет `std::atomic`, но при тестировании он будет подменяться специальной реализацией `FlakyAtomic`, которая будет провоцировать дополнительные переключения потоков при исполнении.