Лабораторная работа №4.
Сделать любой из двух вариантов на выбор, на любом языке. Первый вариант попроще (попросту поэкспериментировать в контролируемой среде), второй поинтереснее.

Вариант 1.
Спровоцировать и пронаблюдать проблемы, возникающие в многопоточных программах при отсутствии синхронизации потоков или неправильном её выполнении.

1. Создать программу из пяти потоков — основного, A, B, C и D:
    1.1. Основной поток запускает остальные четыре и ожидает их завершения.
    1.2. Поток A с определёнными временными интервалами добавляет в список S числа 1, 2, 3 и т. д.
    1.3. Поток B извлекает из списка S последний элемент, возводит его в квадрат и помещает в список R. Если в списке S нет элементов, поток B ожидает одну секунду функцией Sleep().
    1.4. Поток C извлекает из списка S последний элемент, делит его на 3 и помещает в список R. Если в списке S нет элементов, поток C ожидает одну секунду.
    1.5. Поток D извлекает из списка R последний элемент и печатает его. Если в списке R нет элементов, поток D печатает сообщение об этом и ожидает одну секунду.

Синхронизацию потоков производить на данном этапе не нужно.
Запустить программу несколько раз, пронаблюдать результаты и занести их в отчет. Стабильной работы программы, очевидно, не ожидается.

2. Обеспечить корректную синхронизацию потоков.
    2.1. Каждое обращение к спискам S и R из любого потока защищать критической областью (два разных объекта синхронизации, по одному на каждый из списков)
    2.2. Запустить программу несколько раз, пронаблюдать результаты, занести в отчет результаты наблюдений.

Ожидается, что программа будет работать стабильно, не только не завершаясь аварийным образом, но и не «зависая».

3. Спровоцировать взаимоблокировку вследствие состязания.
    3.1. Изменить алгоритм потока B на следующий:
    1) вход в критическую область для списка S, вход в критическую область для списка R;
    2) извлечение элемента из S, вычисление, добавление элемента в R;
    3) выход из критической области для списка R, выход из критической области для списка S.
    3.2. Аналогично изменить и алгоритм потока C, но порядок входа и выхода из критических областей.
    3.3. Повторить пункт 2.2.

В целях наглядности можно между обращениями к КО в обоих потоках вставить задержки функцией Sleep(). Так можно проверить сценарии с разным порядком входа и выхода из КО. Результаты внести в отчёт и пояснить.

Ожидается зависание потоков A, B и C, что будет проявляться в постоянно пустом списке R.

4. Спровоцировать блокировку при неконтролируемом удержании КО.

4.1. В версии программы, полученной в п. 2, внести изменение в алгоритм потока B: после захвата критической области для списка R с некоторой вероятностью следует произвести аварийное завершение работы потока.

4.2. Запустить программу, занести в отчет и объяснить результаты.

Вариант 2.

Задача о пяти обедающих философах.

Суть задачи следующая. Пять философов сидят за круглым столом. Они проводят жизнь, чередуя приёмы пищи и размышления. В центра стола находится большое блюдо спагетти. Чтобы съесть порцию, каждому философу нужно две вилки. Однако, вилок всего пять: между каждой парой рядом сидящих философов лежат по одной вилке, и каждый философ может пользоваться только теми вилками, которые лежат рядом с ним, слева и справа. Философ не может брать две вилки одновременно: сначала он тратит некоторое время на то, чтобы взять одну, затем вторую. Однако, он может одновременно положить их на место.

Задача заключается в том, чтобы написать программу, моделирующую поведение философов. Очевидно, что раз вилок всего пять, то одновременно есть могут не более двух философов, и два сидящих рядом философа не могут есть одновременно. Для имитации периодов раздумий и приёмов пищи можно использовать генератор случайных чисел, позволяющий задавать времена их действий в определённом интервале. Имитация поведения каждого философа, по сути, разбивается на то, что в любой момент времени философ находится в одном из пяти состояний: размышляет, берёт левую вилку, берёт правую вилку, ест, кладёт вилки на место. Таким образом, вилки являются разделяемым ресурсом.

На программу накладываются условия:

1. Каждый философ, по сути, является потоком, и модель поведения у каждого из них должна быть одинаковой, кроме того, какие вилки они могут брать.
2. Накладывание блокировки по сути является действием по взятию вилки, поэтому накладывать блокировку сразу на обе вилки нельзя; последовательность действий должна быть «наложить блокировку – взять вилку – наложить вторую блокировку – взять вторую вилку».
3. Программа должна избегать ситуации взаимоблокировки: ситуации, в которой все философы голодны, то есть ни один из них не может взять себе две вилки (например, когда каждый держит по одной и не хочет её отдавать).

Запрограммировать остановку алгоритма по достижении контрольного времени (например, атомарной операцией над булевым флагом). В отчёте построить некоторый результат работы алгоритма, которая может быть в виде графика, таблицы, лога или

чего угодно ещё; главное условие состоит в том, чтобы по результатам можно было однозначно определить, чем в каждый момент времени был занят каждый философ (одно из пяти состояний).