1. В чем недостаток организации взаимоисключения работы критических секций на основе использования блокирующей глобальной переменной (замка)? Для приведенного алгоритма опишите ситуацию, когда его использование будет приводить к нежелательному развитию событий (в чем будет проблема)

```
shared int lock = 0;

    while(lock); lock = 1;

    critical section

    lock = 0;
```

Недостаток такого способа в том, что условие взаимоисключения может быть нарушено. Например: один из процессов дождался освобождения замка while(lock) и после этого был прерван вторым процессом. Второй процесс зашел в критическую секцию и был прерван первым процессом во время ее выполнения. Первый процесс начал также выполнять критическую секцию. Тогда замок будет как бы закрыт обоими процессами и они оба будут в критической секции, т.е. разделять неразделяемый ресурс.

2. В каком случае целесообразно использовать алгоритм строгого чередования, приведенный ниже, а в каком случае его использование невозможно?

```
shared int turn = 0;

    while(turn != i);

        critical section

    turn = 1-i;
```

Алгоритм строгого чередования целесообразно использовать в тех случаях, когда мы точно знаем, что всем участвующим процессам в текущее время нужен этот ресурс. Если же какой-то процесс спит или не хочет использовать ресурс, то он не сможет передать управление другому процессу, которому этот ресурс нужен.

3. Какого рода проблемы и в какой момент исполнения кода могут возникнуть при использовании приведенного ниже алгоритма флагов готовности?

```
shared int ready[2] = (0, 0);

    ready[i] = 1;

    while(ready[1-i]);

        critical section

    ready[i] = 0;
```

Может возникнуть проблема тупика. Пусть после установления флага готовности первого процесса его прервал второй процесс. Тогда второй процесс также установит флаг готовности и будет ждать, пока первый перестанет быть готовым. Пусть его прервал первый процесс, и он тоже будет ждать, пока второй перестанет быть готовым. В итоге они будут бесконечно ждать друг друга.

4. Сформулируйте формальные определения операций P(S) и V(S) над переменной-семафором S.

P(S): Операция уменьшения. Если значение семафора S больше 0, то уменьшаем его на 1 и продолжаем выполнение программы. В противном случае поток блокируется до тех пор, пока значение S не станет положительным.

V(S): Операция увеличения. Увеличивает значение семафора S на 1. Если есть ожидающие потоки, один из них разблокируется.

5. Перечислите условия возникновения тупиков.

1. Взаимоисключение

2. Ожидание ресурса

3. Неперераспределяемость

4. Круговое ожидание