**Task 2 (6 marks)**
**Serialization graph testing, 2PL, and Timestamp ordering scheduler**

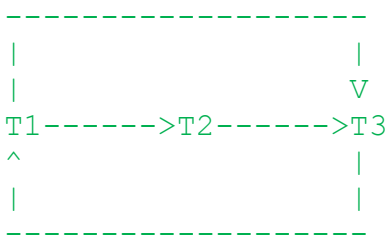
Consider a concurrent execution of database transactions `T1`, `T2`, and `T3` such that the execution is not controlled by any scheduler.

```
T1                T2                T3
read(x)
write(x,x+1)
                  read(y)
                  write(x,y+1)
                                    read(z)
                                    write(x,z+1)
read(z)
write(z,x+2)
```

(1) (2 marks)
Assume, that the transactions attempt to interleave their operations in the same way as in the execution above. Show a sample concurrent execution of the transactions `T1`, `T2`, and `T3` that is controlled by serialization graph testing scheduler.

```
T1                T2                T3
read(x)
write(x,x+1)
                  read(y)
                  write(x,y+1)
                                    read(z)
                                    write(x,z+1)
read(z)
write(z,x+2)
abort
                  commit
                                    commit
```

Draw a conflict serialization graph.




When visualizing the concurrent executions use a technique of two-dimensional diagrams presented to you during the lecture classes, for example, see a presentation 10 Introduction to Transaction Processing (1), slide 9.

(2) (2 marks)
Assume, that the transactions attempt to interleave their operations in the same way as in the execution above. Show a sample concurrent execution of the transactions `T1`, `T2`, and `T3` that is controlled by 2PL scheduler.

Assume, that to simplify the problem we use only a general concept of a `lock` and we do not distinguish between `shared locks` and `exclusive locks`.

When visualizing the concurrent executions use a technique of two-dimensional diagrams presented to you during the lecture classes, for example, see a presentation 10 Introduction to Transaction Processing (1), slide 9.

```
T1                T2                T3
lock(x)
read(x)
write(x,x+1)
                  lock(y)
                  read(y)
                  lock(x)  wait
                                    lock(z)
                                    read(z)
                                    lock(x)  wait
lock(z)wait
abort:deadlock
                                    lock(x)
                                    write(x,z+1)
                                    unlock(z)
                                    unlock(x)
                  lock(x)
                  write(x,y+1)
                  commit
                                    commit
```

(3) (2 marks)
Assume, that the transactions attempt to interleave their operations in the same way as in the execution above. Show a sample concurrent execution of the transactions `T1`, `T2`, and `T3` that is controlled by timestamp ordering scheduler.

When visualizing the concurrent executions use a technique of two-dimensional diagrams presented to you during the lecture classes, for example, see a presentation 10 Introduction to Transaction Processing (1), slide 9.

Show the data items accessed by the transactions together with the timestamps left by the transactions on the data items.

| | | | persistent | | |
|---|---|---|---|---|---|
| T1 | T2 | T3 | x | y | z |
| timestamp(t1) | | | | | |
| read(x) | | | r:t1 | | |
| write(x,x+1) | | | w:t1 | | |
| | timestamp(t2) | | | | |
| | read(y) | | | r:t2 | |
| | write(x,y+1) | | w:t2 | | |
| | | timestamp(t3) | | | |
| | | read(z) | | | r:t3 |
| | | write(x,z+1) | w:t3 | | |
| read(z) | | | | | r:t1 |
| write(z,x+2) | | | | | w:t1 |
| abort | | | | | |
| | commit | | | | |
| | | commit | | | |