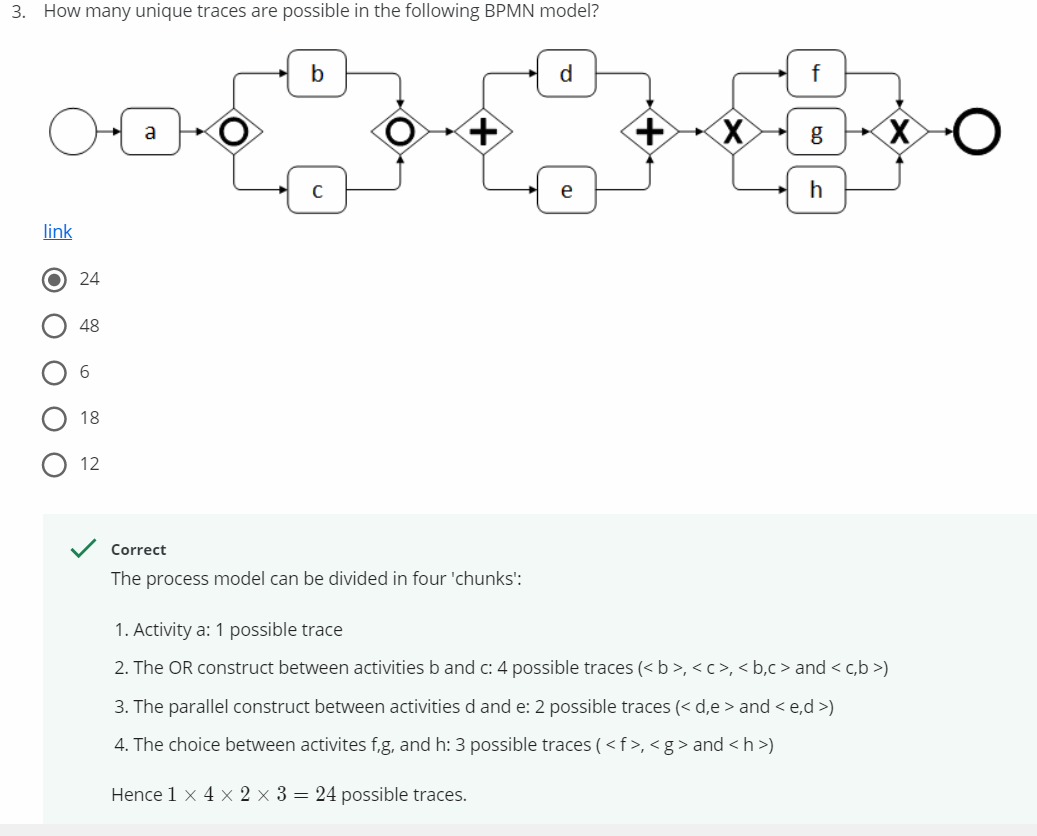
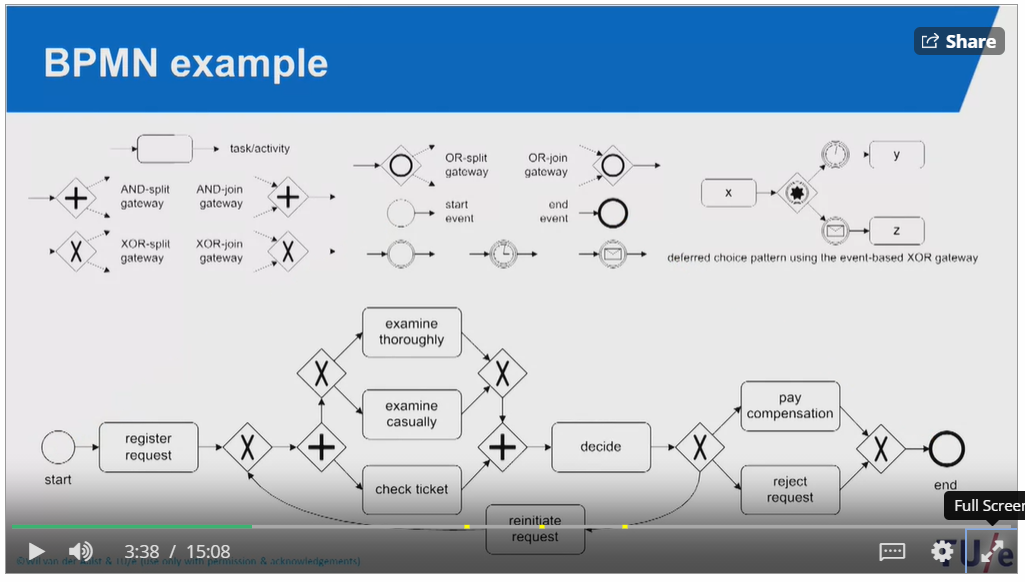


First Option: Wrong; <a,b,c,d,e,g> might be possible scenario which is not present in event log

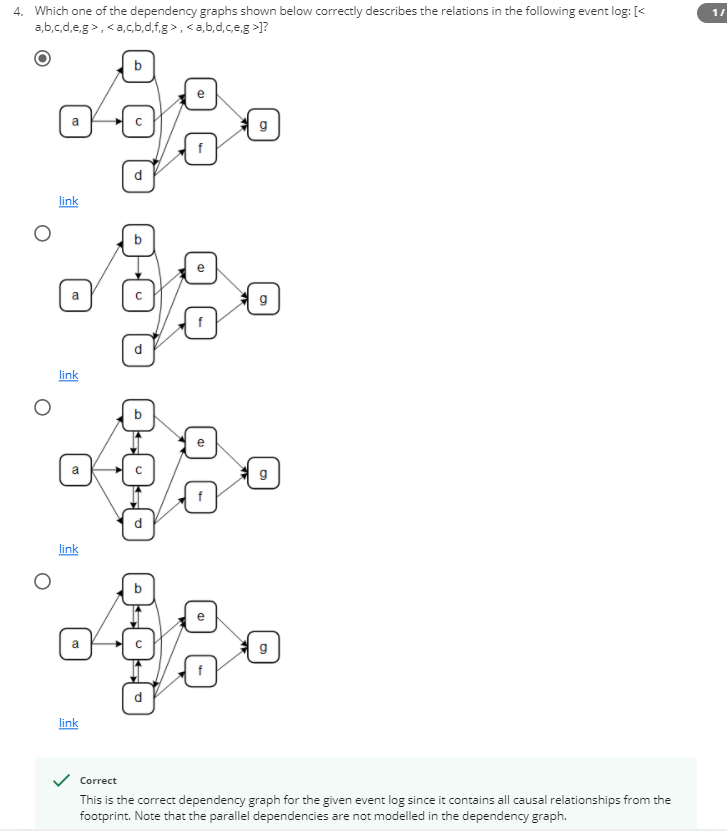
Second Option: Wrong; Due to having many arrows at e and f, it may lead to variety of possibilities like <a,c,b,d,e,f>

Third Option: Correct; It fits to the question.





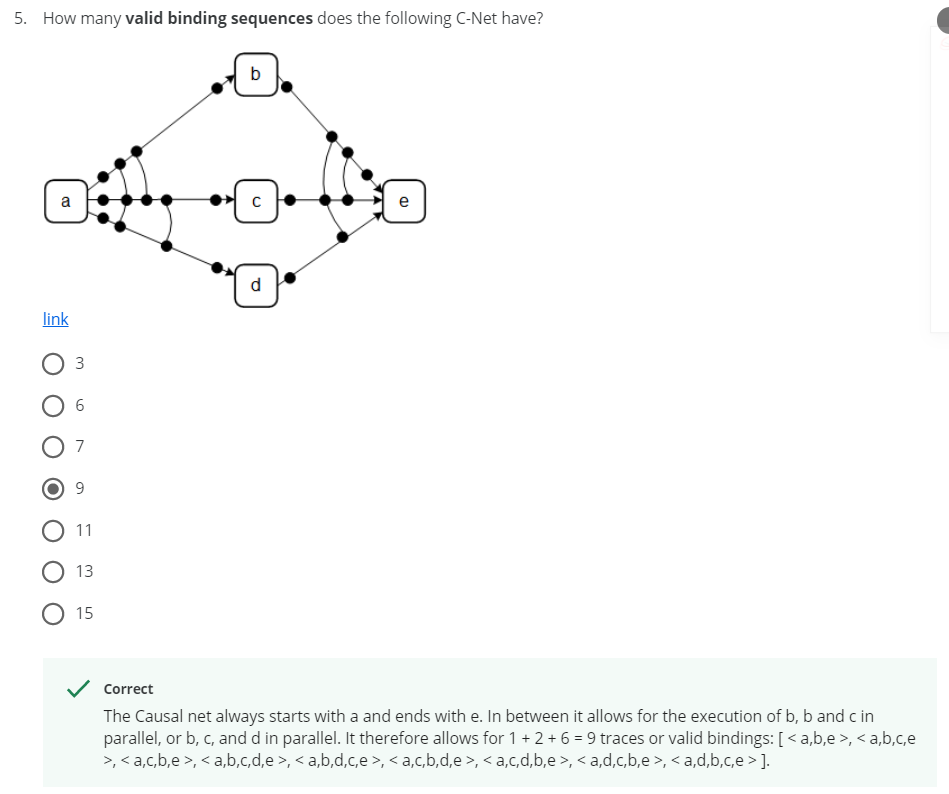
Circle Symbol - Any possibility can happen(even multiple activities at a time)



Parallel dependencies are not modelled in the dependency graph leaves 3rd and 4th option to recycle bin

2nd option - <a,c,b,d,f,g> is not possible

1st option - Satisfies event log

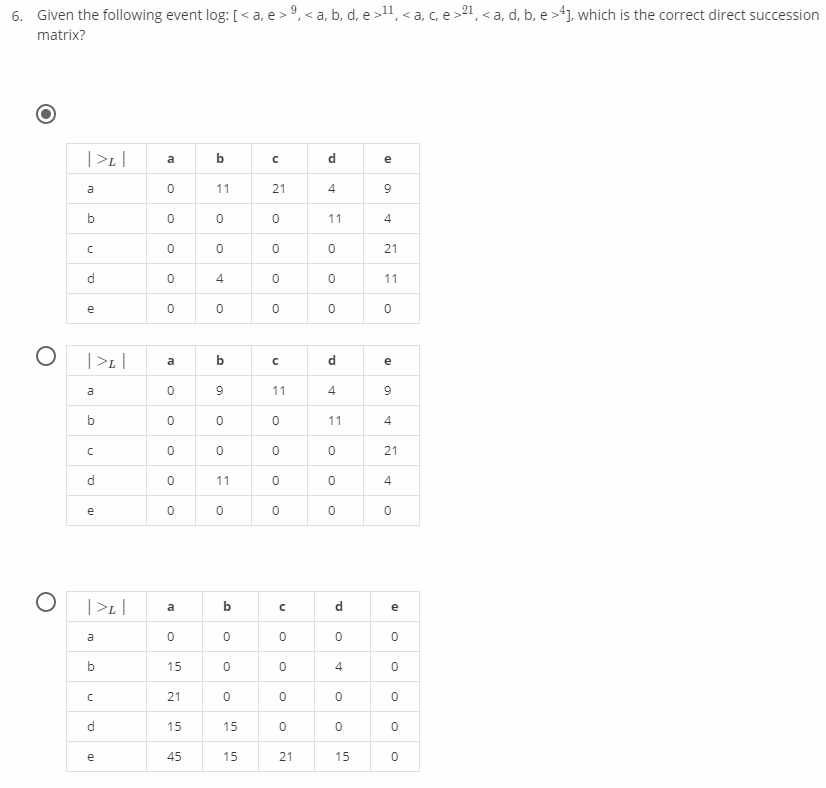


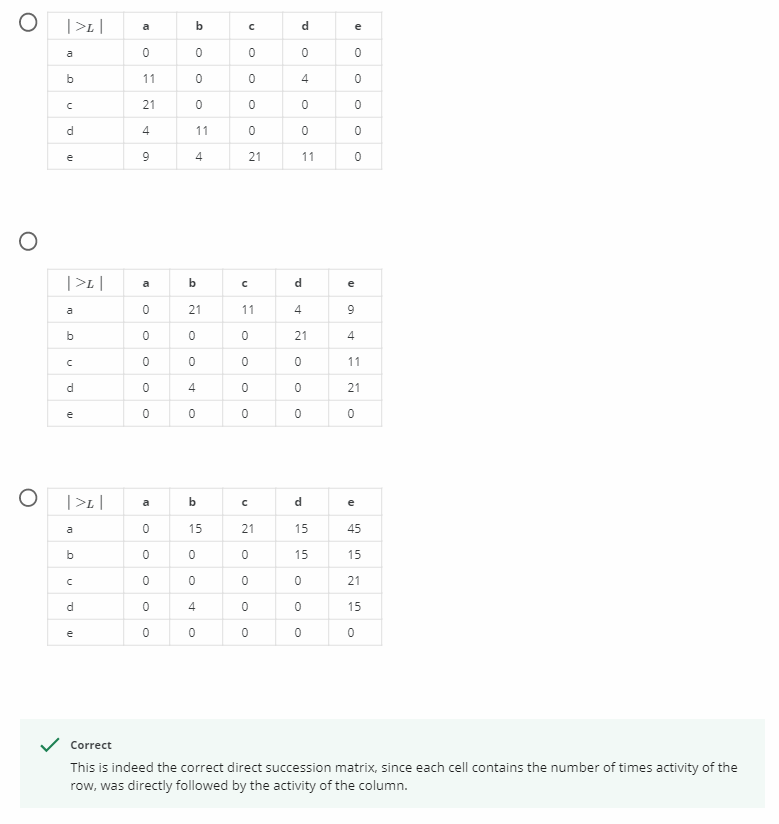
For single dots: Possible dots at input and output is a binds e and b and e => 1

For double binding: Possible double binding at input and output is b and c => 2

For triple binding: Possible triple binding at input and output is a, b and c => 6

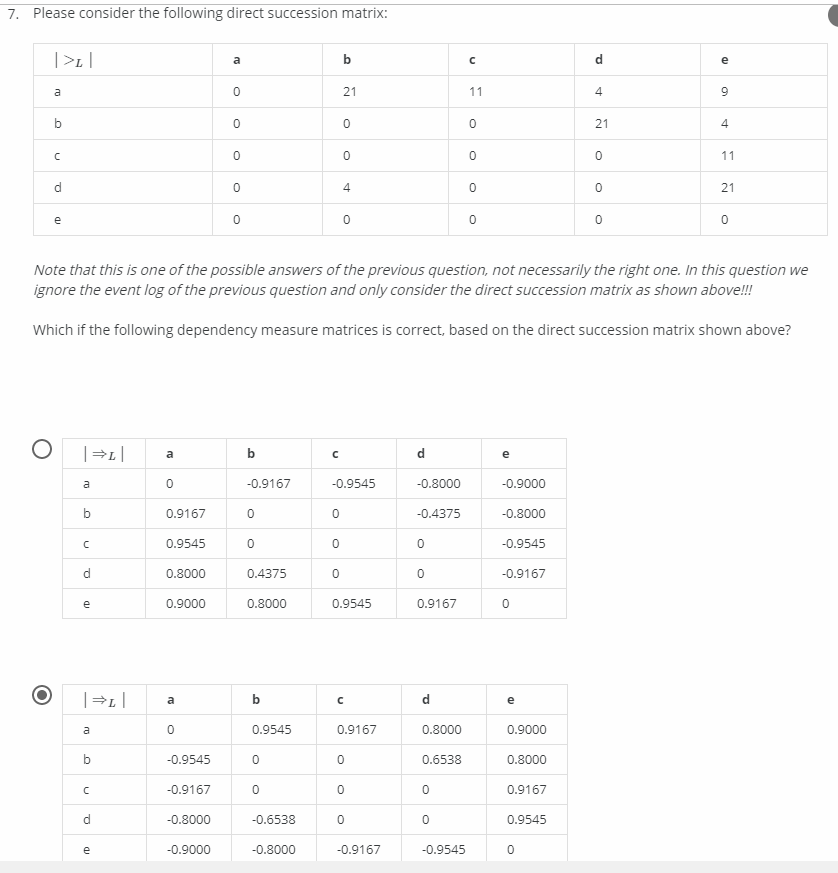
Total = 9

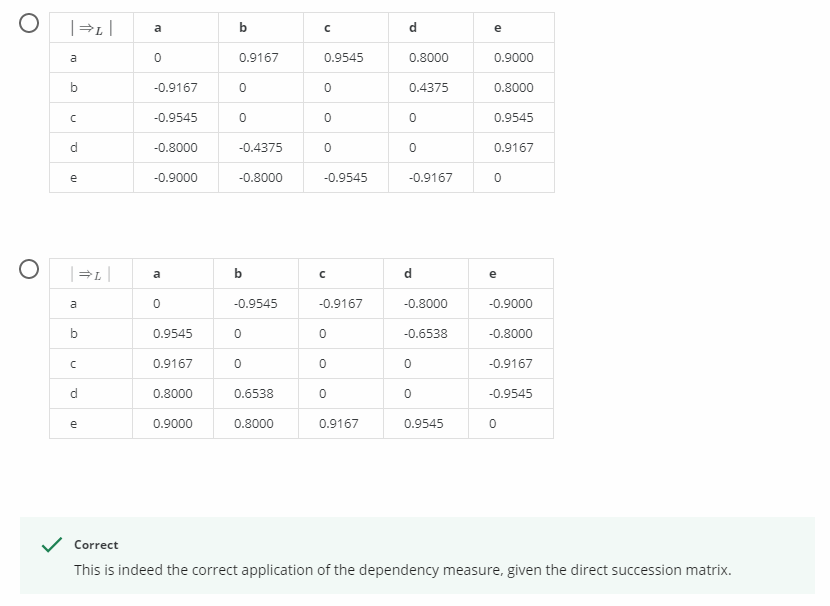


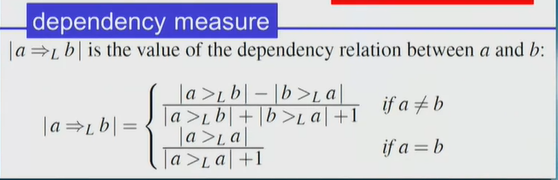


It’s just counting number of times an activity follows another activity

Like for a and b: <a,b,d,e>11 => so 11 times





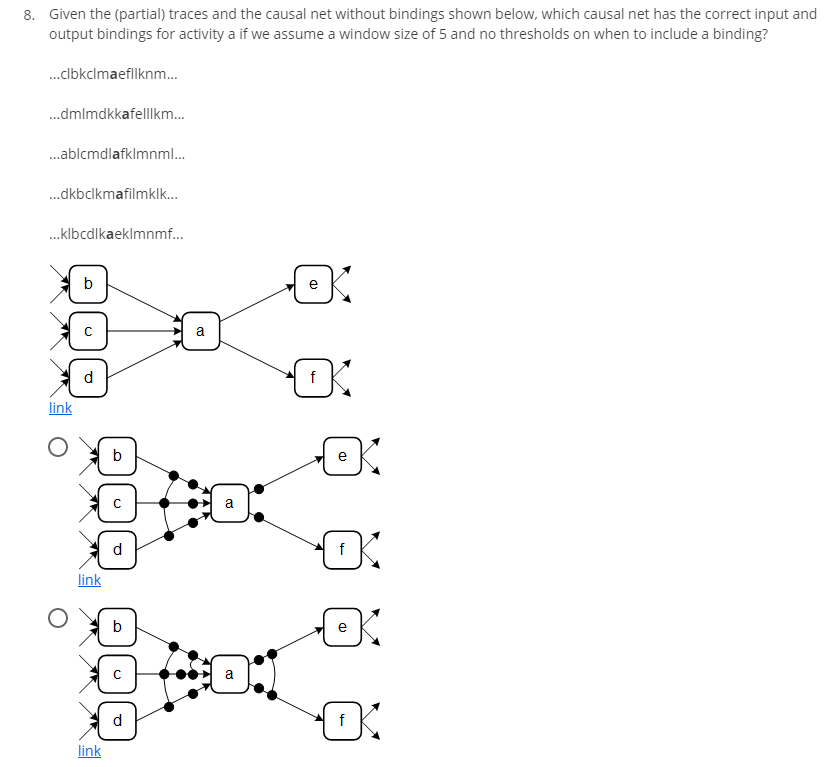


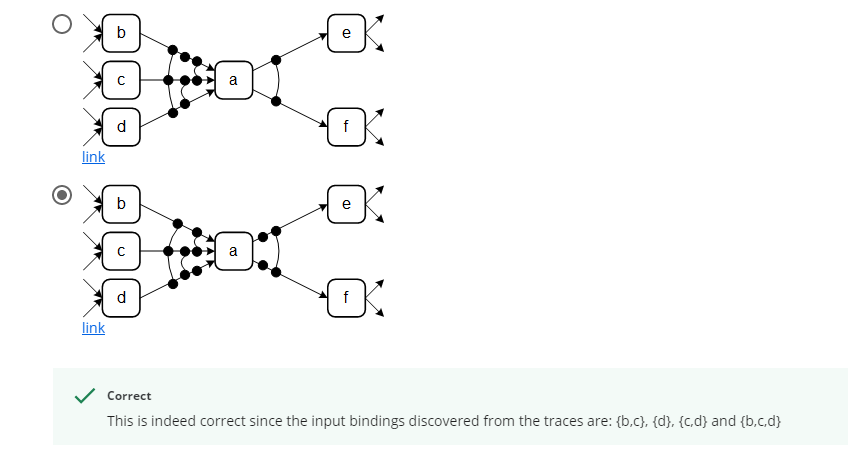
For a>La => We will apply second formula = 0/(0+1) = 1

For a>Lb => We will apply first formula as a not = b = (21-0)/(21+0+1) = 0.9545

And so on for all possibilities in the table

2nd option satisfies





Window size: 5 => consider 5 activities before and after ‘a’

bkclm**a**efllk => input(bc), output(ef)

lmdkk**a**felll => input(d), output(ef)

similarly => input(cd), output(f)

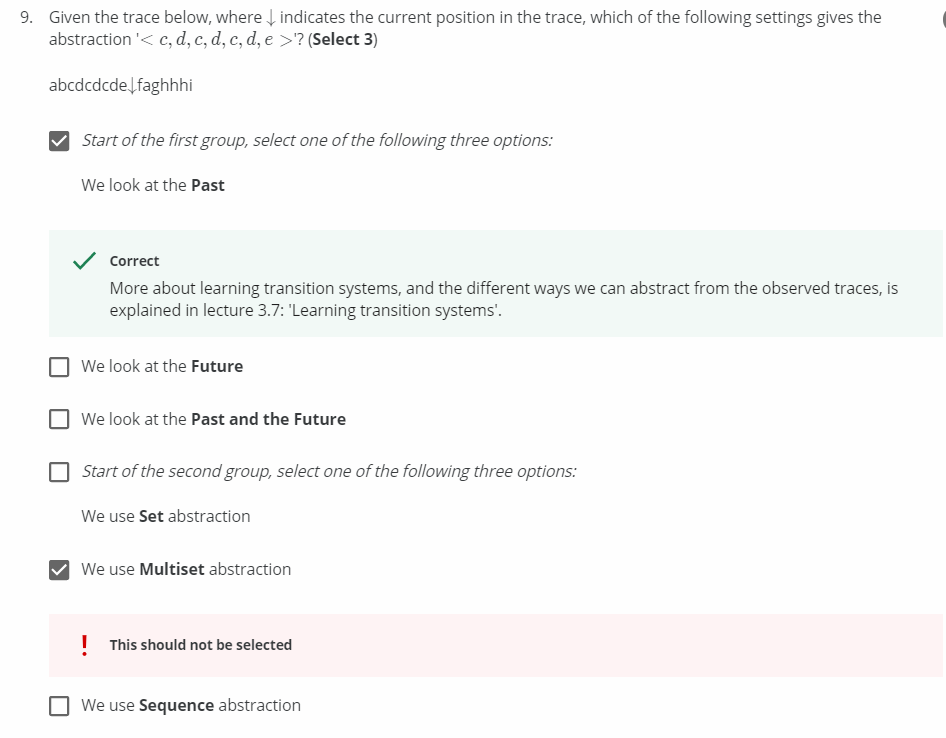
* input(bc), output(f)
* input(bcd), output(e)

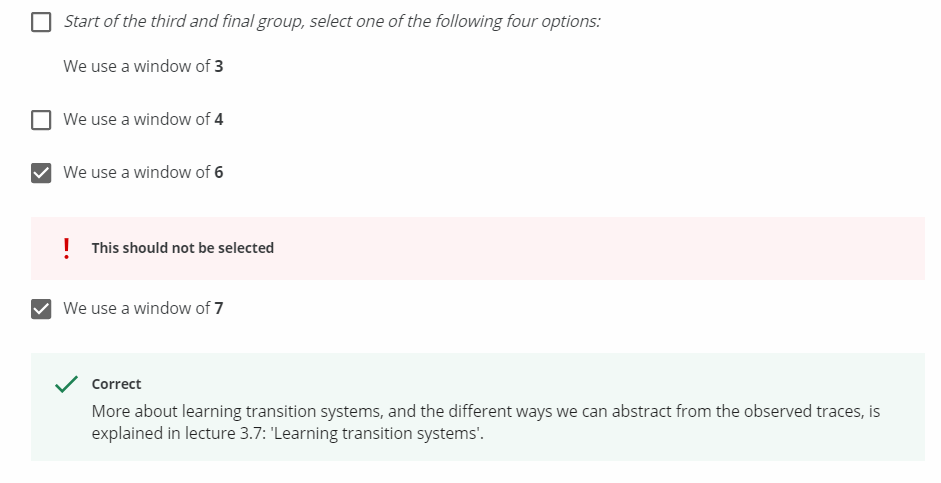
Single binding - d,e,f

Double binding - bc,cd,ef

Triple binding - bcd

Last option satisfies this.





Left side activities from the arrow => Past

Right side activities from the arrow => Future

For first group:

Given abstraction <c,d,c,d,c,d,e>is present in the left side of the arrow(Past)

For second group:

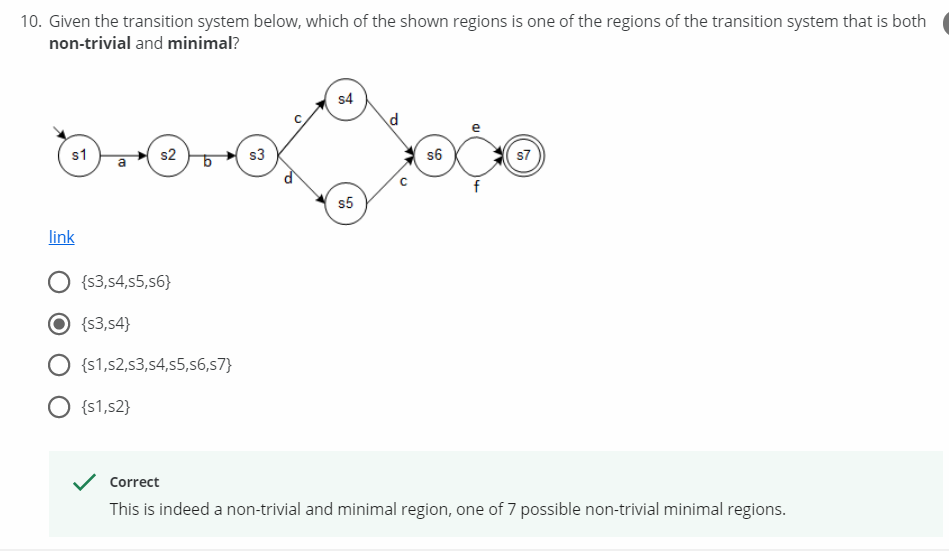
It followed Sequence abstraction by the checking the event log

Not set abstraction, output will be <c,d,e>

Not multiset abstraction, output will be <c3,d3,e>

For third group:

Smallest size out of past and future



For s3 and s4,

a does not cross

d exits

c does not cross

b enters

e does not cross

f does not cross