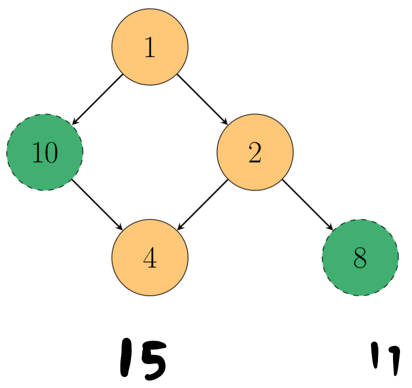
1.) If there is a very long list of items that need to be read and only wrote to once

Is when a readers-writers lock is better than a regular mutex.

This is because a R-W lock lets there be multiple reads and no writes. When all the reads are done it can then easily do the one write since the second option allows no reads and a single write

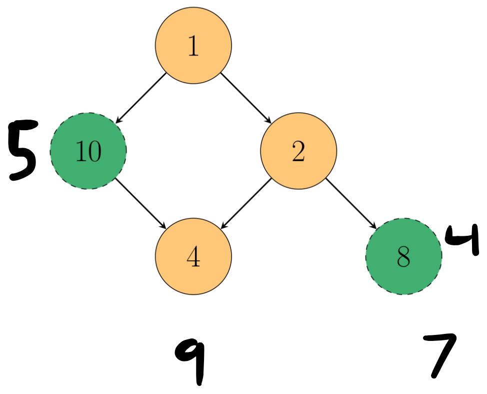
2.)

a.)



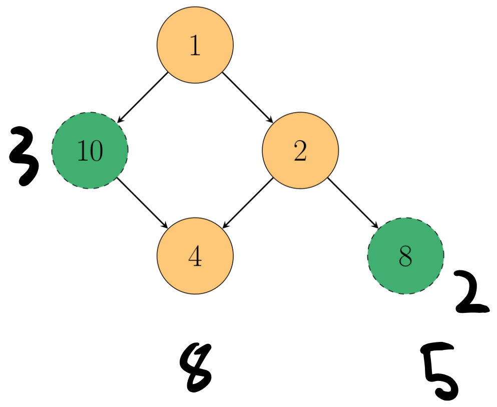
$\boxed{15}$

b.



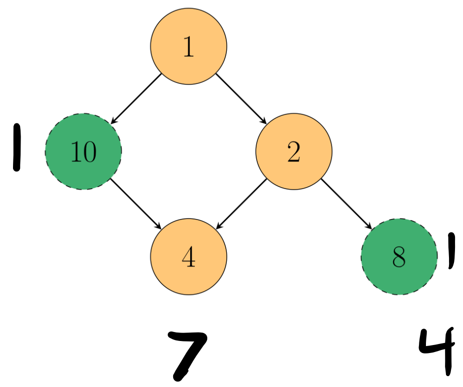
$\boxed{9}$

c.



$\boxed{8}$

D.



$\boxed{7}$

3.) mean

```
Def sum_range( A, start, end, Queue)
 Sum=0
 for i in range(start, end)

    sum ± A[i]
queue.put(sum)



Def main

 processes=[]
 Queue = Queue()

 For i inrange(num_processors)

    Start = i * len(A) // num_processors
    end  =(i + 1) * len(A) // num_processors

    Process=(target=sum_range(A, start, end,
                                queue)
    Processes.append(process)

 for processin Prosseses:
      process.start()

 for process in processes:
      process.join()
```

```
sum = 0
for i inrange(num processes)
    sum I Queue.get()

return sum
```

compelexity $= \left(\frac{n}{p}\right) + p = \left(\frac{n}{p}\right)$

B.) mode

```
Def Use_dict(A , Start, end , Dict):
    For i inrange(Start, end)
        Dict[A[i]] ± 1



Def main
    Dict = { 0:0 , 1:0 , 2:0
            3:0 , 4:0 , 5:0
            6:0 7:0 8:0
            9:0 }
    For i inrange(num_Processors)
        Start = i * len(A) // num_processors
        end = (i + 1) * len(A) // num_processors
        Process = (target= Use_Dict (A , Start, end, Dict)
        Processes.append(Process)

    For process in Prosseses:
        process.Start()

    for process in processes:
        process.join()
```

return (max(Dict.values))

Complexity = $\left(\frac{n}{p}\right)$