ans1) a) speedup $(S) = \dfrac{1}{(1-p) + \frac{p}{n}}$

$p \rightarrow$ parallel part

sequential part (given) = 40% = 0.4.

p = 1 − sequential part = 0.6.

$$\Rightarrow S = \frac{1}{0.4 + \frac{0.6}{n}} \quad ; \quad \lim_{n \to \infty} \frac{0.6}{n} = 0$$

$$\Rightarrow S(\text{max. possible}) = \frac{1}{0.4} = \underline{\underline{2.5}}$$

b) ~~the~~ $S_n' = \dfrac{1}{\frac{(1-p)}{k} + \frac{p}{n}}$; $S_n = \dfrac{1}{(1-p) + \frac{p}{n}}$

here, it is given that $p = 1 - \frac{30}{100} = 0.7$

and as sequential part of S_n' is speedup by k it becomes $\left(\frac{1-p}{k}\right)$.

Now, $S_n' > 2 S_n$

$$1 - p + \frac{p}{n} > 2\left(\frac{1-p}{k} + \frac{p}{n}\right)$$

$$\Rightarrow 0.3 + \frac{0.7}{n} > \frac{0.6}{k} + \frac{1.4}{n}$$

$$\Rightarrow 0.3 - \frac{0.7}{n} > \frac{0.6}{k}$$

$$\Rightarrow k > \frac{0.6}{\left(0.3 - \frac{0.7}{n}\right)}$$

c) Given, $S' = 2S$

where, $S = \frac{1}{1-p+\frac{p}{n}}$, $S' = \frac{1}{\frac{1-p}{3}+\frac{p}{n}}$.

since in S', there was a speedup of factor 3, time taken would be $\frac{1}{3}$ times that of original

$$\Rightarrow 1-p+\frac{p}{n} = \frac{2-2p}{3} + \frac{2p}{n}.$$

$$\Rightarrow \frac{1}{3}(1-p) = \frac{p}{n}.$$

$$\Rightarrow n - np = 3p \Rightarrow p(3+n) = n$$

$$\Rightarrow p = \frac{n}{3+n}.$$

2) a) Mutual Exclusion:

Assume that the Peterson tree lock doesn't satisfy mutual exclusion. This means that at some time t, two threads have gotten access to the CS. In this case, in the binary tree of peterson locks there is some ~~node~~ node (Peterson 2-proc lock) ~~where~~ which allowed two different threads to pass. as of time 't'.

But, we ~~already~~ already know that peterson 2-process algo ~~is~~ satisfies mutual exclusion so our assumption gave us a contradiction. Hence, Peterson tree lock satisfies mutual exclusion

Starvation free

Let us assume starvation exists, then a thread (p) is stuck at the line where busy waiting happens.. ~~This~~ Let this occur at lowest level from the ~~top~~ bottom. Now each node is a peterson 2 process lock. Let this node be locked by (q) which made p to be stuck. But at some point of time, as we already know each node individually is starvation free, ~~and~~ q will enter CS eventually & while exited allow p to proceed because of the property of starvation free of peterson 2-process lock. But this is contradiction that p is stuck forever. Hence it is ~~starvation~~ free.

Also, lower bound for waiting would be:
Let i is total level & 2^i is nodes in binary tree.
A thread has to wait for its sibling to gain access to CS before proceeding. So, by recurrance:

$$f(i) = f(i-1) + f(i-1) + 1$$

(for sibling)

Also $f(2) = 1$ (as we know peterson 2-proc ~~node~~ lock is starvation free)

$f(i) = 2^{i-2} + 2^{i-2} - 1$

$\Rightarrow$ on solving, we get $f(i) \sim O(n)$ @ $O(2^i)$

& we found a bound, it is starvation free.

Starvation free implies deadlock free

3) When there is no contention, the algorithm is indeed fast as, "x == i" when thread executes #10. which lets thread into CS without the need for taking lock which is usually time taking.

But on contention, this algorithm violates mutual exclusion.

Let us say two threads A and B are trying to get access to CS.

If both of the threads A and B executes all statements till 8. only (no one executes 9 yet), then at this point variable x contains either "A" or "B". (thread ID. (the last one to execute #7).

After they both execute #9, the variable "y" similar to previous case will have either "A" or "B" thread ID which doesn't matter to us anyway.

But at #10, without loss of

generality assume x has thread A's ID.
A will ~~attain~~ skip #11 and go into the CS.
B will execute #11 and attains the lock. But now, there is no guarantee that A has exited CS, so now both A and B are in CS. Mutual exclusion is violated.