(a) No, there are cases where there is no solution.
For example, consider the graph
G = (1)–(2)–(3)
where (1), (2), and (3) are the three vertices of the graph and – represents an edge connecting two vertices.
Suppose L = {(1), (2), (3)} then there exists no spanning tree T for G where nodes (1), (2), and (3) are all leaves because one of the nodes must connect to **two** of the other nodes for the tree to remain connected.

(b) MST(V, E, L)

```
1  for i = 1,2,...n      n is the number of vertices in V
2      if v_i is in L
3          tempE = []
4          for j = 1,2,...,m      m is the number of edges in E
5              if e_j has v_i as one of its endpoints
6                  tempE = tempe ∪ e_j
7      sort(tempE)
8      remove from E all edges in tempE except for the one with the smallest weight
9  kruskal(V, E)
```

(c) **This algorithm works based on the following intuition:**
Since each leaf node can only have one connection, get rid of the edges in E that connect this leaf node to another node provided that there are shorter edges that can connect this leaf node to another node. If there is no shorter edge, keep the edge. We can safely do this because we know that a leaf vertex can only have one connection (so make sure the connection is the shortest of all possible connections).
After this is done we can simply run any algorithm we have learned in class on our (possibly) modified inputs to get a result that will always satisfy our addition constraints (since there is no way for a leaf node to have more than one possible connection).