Problem 1

Iteration (i)	e_i	$c(e_i)$	**Edges in T at the end of iteration i**	**Is T disconnected?**
1	(D,E)	13	(A,B),(A,C),(B,C),(B,E),(C,D),(C,F),(D,F)	No
2	(C,D)	11	(A,B),(A,C),(B,C),(B,E),(C,F),(D,F)	No
3	(A,C)	10	(A,B),(B,C),(B,E),(C,F),(D,F)	No
4	(B,C)	9	(A,B),(B,C),(B,E),(C,F),(D,F)	Yes
5	(B,E)	8	(A,B),(B,C),(B,E),(C,F),(D,F)	Yes
6	(C,F)	7	(A,B),(B,C),(B,E),(C,F),(D,F)	Yes
7	(A,B)	6	(A,B),(B,C),(B,E),(C,F),(D,F)	Yes
8	(D,F)	5	(A,B),(B,C),(B,E),(C,F),(D,F)	Yes

Problem 2

(a) We can update the original MST by first adding the lightest edge from the new edges connected to the newly introduced node. After that, we check the remaining new edges. Note that adding any of these edges will cause a cycle to form in the MST. For each of these edges, we identify the cycle formed when the edge is added and iteratively remove the edge with the heaviest weight from the cycle. The following pseudocode can be used:

```
$v \leftarrow$ new node
**Function** Update MST (*new node* $v$, *graph* $G(V,E,W)$, *graph* $T(V',E',W')$)
Add the edge $(v,u)$ with minimum weight to $E'$
**foreach** $(v,u)$ in $E$ not in $E'$ **do**
    Add $(v,u)$ to $E'$
    Identify the heaviest edge in the cycle containing $(v,u)$
    Remove the heaviest edge from $E'$
**end foreach**
return $T$
```

(b) Since the MST follows a tree structure, if a cable is damaged, the MST will split into two disconnected components. The most efficient way to update and reconstruct the MST is to reconnect the two components by finding and adding the lightest edge between them:

```
**Function** Update MST (*removed edge* $e$, *graph* $G(V,E,W)$, *graph* $T(V',E',W')$)
$C1(U1,R1), C2(U2,R2) \leftarrow$ disconnected graphs resulted by removing $e$
$Min\_weight = infinity$
$New\_edge = None$
**foreach** $(w,u) \in E$ such that $w \in U1$ **AND** $u \in U2$ **do**
    **if** $(w,u)$.weight$<$ $Min\_weight$ **then**
        $Min\_weight = (w,u)$.weight
        $New\_edge = (w,u)$
    **end if**
**end foreach**
Add $New\_edge$ to $E'$
return $T$
```

(c) We again examine the cycle formed by adding the newly updated edge and remove the heaviest edge in that cycle:

```
**Function** Update MST (*replaced edge* $e$, *graph* $G(V, E, W)$, *graph* $T(V', E', W')$)
Add $e$ to $E'$
$C \leftarrow$ edges in the cycle containing $e$
$Max\_weight = 0$
$Removed\_edge = None$
**foreach** $(w, u) \in C$ **do**
    **if** $(w, u)$.weight$>$ $Max\_weight$ **then**
        $Max\_weight = (w, u)$.weight
        $Removed\_edge = (w, u)$
    **end if**
**end foreach**
Remove $Removed\_edge$ from $E'$
return $T$
```

Problem 3

We will do a proof by contradiction, showing that if T' is not the minimum spanning tree of G', then there must be a different minimum spanning tree of G'.

Suppose T' is not an MST of G'.

It is given that T' is connected.

Given the definition of a minimum spanning tree, T' must

- include all vertices of G'
- be acyclic
- be connected.

Therefore, G' must also be connected.

Since we are supposing that T' is not an MST of G', there must exist a separate spanning tree, say T^*, for graph G'.

$$\text{weight}(T^* \cup (T \setminus T')) < \text{weight}(T)$$

In other words, if you replace T' with T^* for the minimum spanning tree, it should be cheaper (as T' is supposedly costlier than T^*)

Given the fact that T' is induced by a subset V' of original vertices V, T' includes the same edges as the subset of vertices from T.

Since T' spans the same vertices as T, it must have the lowest edge weights, as T is the MST for G.

Also note that in the case where the subset of vertices is not connected by edges in the original MST T, but by costlier edges, the lowest weight edge connects the nodes by blue rule.

Given the definition of an MST having the lowest edge weights, T' must have the lowest edge weights, which contradicts the assumption that T^* has lower edge weights than T'.

Therefore, T' must be the MST of G' (rather than another spanning tree, T^*).

Problem 4

Minimizing the product of the edge costs is equivalent to minimizing the sum of the *log* of the costs:

$$min \prod_{e \in T} c_e \leftrightarrow min\ log(\prod_{e \in T} c_e) \leftrightarrow min \sum_{e \in T} log(c_e)$$

Therefore, to find the tree that minimizes the product of the edge costs, we can apply the logarithm transformation to the edge costs, which converts the problem into the standard MST objective. Then, we can use any MST algorithm to find the optimal tree:

```
Function Prim_MST_Log (graph $G(V, E, W)$)
$V'$ ← set containing the MST nodes
$E'$ ← set containing the MST edges
$Candidate\_edges$ ← list containing the edges incident to $V'$ members
Choose a random node $v_0$ and add it to $V'$
Add the log-transformed edges incident to $v_0$ to $Candidate\_edges$
while $V'$ does not contain all nodes do
    $(u, v)$ ← The edge with the minimum cost from $Candidate\_edges$
    if $v \notin V'$ then
        Add $v$ to $V'$
        Add $(u, v)$ to $E'$
        Add all log-transformed edges incident to $v$ to $Candidate\_edges$
    else
        Remove $(u, v)$ from $Candidate\_edges$
    end if
end while
return $T(V', E')$
```