HW 1

Question 1

Question 2

a) $G = (V, E, w)$

$E \subseteq V \times V$

$w : E \to R$

$V = \{a, b, \ldots\}$

$E = \{e_1 = (a,b), e_2 = (b,c), \ldots\}$

$w(e_1) = 1, w(e_2) = 2 \ldots$

$V = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p\}$

$E = \{e_1 = (a,b), e_2 = (b,c), e_3 = (c,d), e_4 = (d,e), e_4 = (e,f), e_5 = (f,g), e_6 = (g,h),$
$e_7 = (h,i), e_8 = (i,j), e_9 = (j,k), e_{10} = (k,l), e_{11} = (l,m), e_{12} = (m,n)$
$e_{13} = (n,o), e_{14} = (o,p)\}$

$w(e_1) = 1, w$

Question 2

a) $G = (V, E, w)$

$E \subseteq V \times V$

$w : E \to R$

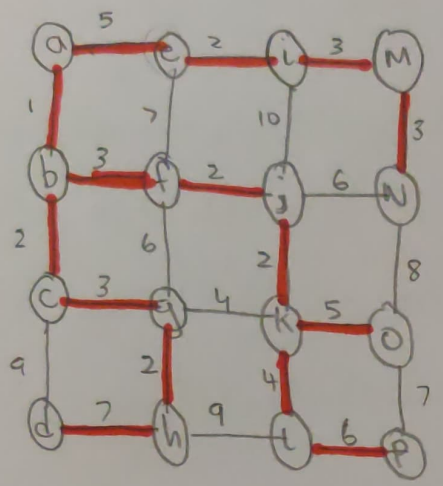
$V = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p\}$

$E = \{ e_1 = (a,b), e_2 = (a,e), e_3 = (b,c), e_4 = (b,f), e_5 = (c,g), e_6 = (c,d), e_7 = (d,h),$
$e_8 = (h,g), e_9 = (h,l), e_{10} = (g,k), e_{11} = (g,f), e_{12} = (f,j), e_{13} = (f,e)$
$e_{14} = (e,i), e_{15} = (i,m), e_{16} = (i,j), e_{17} = (j,n), e_{18} = (j,k), e_{19} = (k,o),$
$e_{20} = (k,l), e_{21} = (l,p), e_{22} = (p,o), e_{23} = (o,n), e_{24} = (n,m) \}$

$w(e_1) = 1, w(e_2) = 5, w(e_3) = 2, w(e_4) = 3, w(e_5) = 3, w(e_6) = 9, w(e_7) = 7,$
$w(e_8) = 2, w(e_9) = 9, w(e_{10}) = 4, w(e_{11}) = 6, w(e_{12}) = 2, w(e_{13}) = 7$
$w(e_{14}) = 2, w(e_{15}) = 3, w(e_{16}) = 10, w(e_{17}) = 6, w(e_{18}) = 2, w(e_{19}) = 5,$
$w(e_{20}) = 4, w(e_{21}) = 6, w(e_{22}) = 7, w(e_{23}) = 8, w(e_{24}) = 3$

b) The challenge in determining the overall cost of a weighted tree, particularly a minimum spanning tree, revolves around identifying a set of edges that links all vertices without forming cycles aiming for the lowest possible cumulative edge weight

c)



(While connecting each vertices)

Total minimum cost

=

50

Question 3

a) t = CTOCOCBCBCTOCTOCBC

P = TOCTOC Σ = {O, B, C, T}

b)

```
  0 1 2 3 4 5
  C T O C O C B C B C T O C T O C B C
3C T O C T O C
 1C  T O C T O C
5C          T O C T O C
 1C           T O C T O C
 6C            T O C T O C
```

SkIP
0
5
1
0

Total comparisons = 3 + 1 + 5 + 1 + 6

= 16

Total alignments = 5

Question 4

Result = "Hello Haskell"

head lst : "H"

tail lst : "ello Haskel"

(head lst : tail lst) ⟶ "Hello Haskell"

tail (head lst : tail lst) ⟶ "ello Haskell"

* head lst : tail (head lst : tail lst) ⟶ "Hello Haskell"