Q1-a

Suppose there are four campers with $w_1 = 1$, $w_2 = 2$, $w_3 = 3$ and $w_4 = 4$.
Assume that C is 5.
Campers 1 and 2 are two lightest campers, since $1 < 2 < 3 < 4$.
$w_1 + w_2 = 1 + 2 = 3 <= 5$, by the algorithm, the pair {1, 2} is added to the result set.
Since no camper can be in two canoes, campers 3 and 4 are two lightest campers in the rest of the campers.
$w_3 + w_4 = 3 + 4 = 7 > 5$, by the algorithm, do nothing to the result set.
Then, the algorithm has covered all the campers. The algorithm stopped.
There is only one pair {1, 2} in the resulting set.
However, if we combined campers 1 and 4 together. Then combine campers 2 and 3 together.
$w_1 + w_4 = 1 + 4 = 5 <= 5$, which satisfies feasible condition (a)
$w_2 + w_3 = 2 + 3 = 5 <= 5$, which satisfies feasible condition (a)
There is no camper appearing in the different sets, which satisfies feasible condition (b).
Therefore, there exists a feasible set with two pairs in it.
Since there exists the feasible set with greater cardinality than the given algorithm, the algorithm in 1-a would not necessarily be an optimal set.

Q1-b

Promising Set Lemma: Let A_i be the set in A at the end of the i-th iteration.
For each iteration i, A_i is contrainted in some optimal set.

Proof: By induction on i.
Basis: for i = 0
At the end of the 0-th iteration of the loop.(i.e. Just before entering the loop for the first time)
A_0 is an empty set, as wanted.

Induction Hypothesis: Assume that A_i is contrainted in an optimal set A*, for all i >= 0.

Induction Step: want to show that A_{i+1} is contrainted in the optimal set Â.
Suppose the **lightest** camper is **p** and **heaviest** camper is **q** in each iteration.

(i) if $w_p + w_q > C$,
then by the algorithm, q is discarded and no pair is added to the result set.
$w_p + w_q > C \Rightarrow w_k + w_q > C$, for any Camper k in the camp [since p is the lightest camper]
⇒ q cannot be paired with any camper in the camp to group a valid canoe.
Therefore, the optimal set would not include Camper q, and it would return the result of the i-th iteration.(i.e. $A_{i+1} = A_i$)
Let's choose Â = A*
A_i is contrainted in optimal set A* [by I.H.]
⇒ A_{i+1} is contrainted in optimal set A* [by algorithm $A_{i+1} = A_i$]
⇒ A_{i+1} is contrainted in optimal set Â [by our choice for Â]
as wanted.

(ii) if $w_p + w_q <= C$,
First want to prove: if the combined weight of the lightest and heaviest camper does not exceed C, there is an optimal set Â that contains a pair consisting of the two campers.

Case 1: A* does not contain p and q.

Since $w_p + w_q <= C$ and p and q do not appear in the optimal set A*, the pair {p, q} needs to be included in the optimal set A* to form a maximum cardinal set, but A* does not contain p and q, which is a contradiction.

Therefore, Case 1 would never happen.

Case 2: A* contains both p and q.

If A* contains (p, q), then choose Â = A*, where Â is an optimal set with pair (p, q).

If p and q appears in A* not in the same pair.

Suppose p is in (p, a) and q is in (b, q).

b and q in the same pair ⇒ $w_b + w_q <= C$

⇒ $w_b + w_a <= C$ [$w_a <= w_q$, since q is the heaviest camper in the camp]

By the condition for case (ii) where $w_p + w_q <= C$,

Campers a, b, p, q can be regrouped into two pairs (p, q) and (a, b) without loss the generalization.

Let Â = A* – {(p, a)} – {(b, q)} ∪ {(a, b)} ∪ {(p, q)}

Â is feasible since A* is feasible and $w_p + w_q <= C$, $w_a + w_b <= C$.

|Â| = |A*| - 2 + 2 = |A*|, so **Â has the same number of pairs as the optimal set A***.

Therefore, Â is an optimal set with the pair {p, q}.

Case 3: A* contains p but does **not contain q**.

Suppose the pair with p in A* is {p, a}.

Let Â = A* – {p, a} ∪ {p, q}

Â is feasible since A* is feasible and q ∉ A* and $w_p + w_q <= C$.

|Â| = |A*| - 1 + 1 = |A*|, so **Â has the same number of pairs as the optimal set A***.

Therefore, Â is an optimal set with pair {p, q}.

Case 4: A* contains q but does **not contain p**.

Suppose the pair with p in A* is {a, q}.

Let Â = A* – {a, q} ∪ {p, q}

Â is feasible since A* is feasible and p ∉ A* and $w_p + w_q <= C$.

|Â| = |A*| - 1 + 1 = |A*|, so **Â has the same number of pairs as the optimal set A***.

Therefore, Â is an optimal set with pair {p, q}.

Therefore, if the combined weight of the lightest and heaviest camper does not exceed C, there is an optimal set Â that contains a pair consisting of the two campers.

⇒ Â contains A_i ∪ {p, q} = A_{i+1} (by construction of the algorithm).

Therefore Â is an optimal set that contains all pairs in A_{i+1}, as wanted.

Q1-c

The algorithm does not work.

Here is a counter example.

Suppose there are eight campers with $w_1= 1$, $w_2= 2$, $w_3= 3$, $w_4= 4$, $w_5= 5$, $w_6= 7$, $w_7= 10$ and $w_8= 11$.

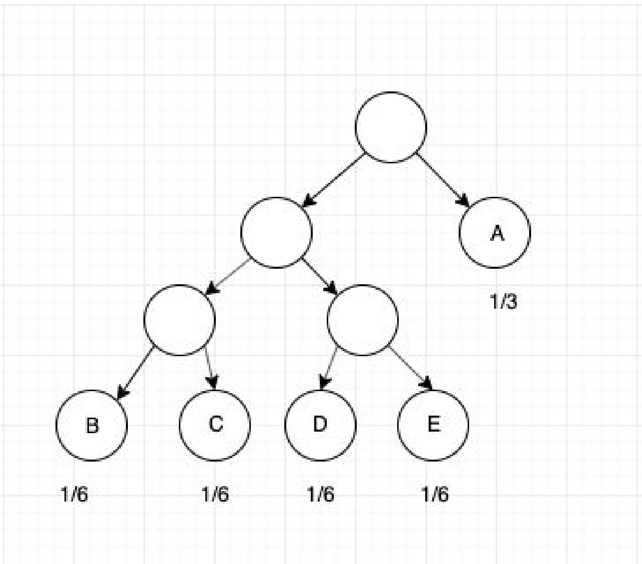
Assume that C is 23.

Two lightest: w_1 and w_2

Two heaviest: w_7 and w_8

Combining weight of two lightest and two heaviest: $w_1 + w_2 + w_7 + w_8 = 1 + 2 + 10 + 11 = 24 > 23$
Then discard the heaviest camper 8.
Recursively applying the algorithm, we would get the final optimal set {(1, 2, 7, 10)} of size 1.
However, if we group campers 1, 3, 6, 8 together and Campers 2, 4, 5, 7 together.
$w_1 + w_3 + w_6 + w_8 = 1 + 3 + 7 + 11 = 22 < 23$
$w_2 + w_4 + w_5 + w_7 = 2 + 4 + 5 + 10 = 21 < 23$
Every camper only appears in one canoe.
Therefore, there exists a set {(1, 3, 6 , 8), (2, 4, 5, 7)} of size two, which contains more groups than the result set of the algorithm.
Hence, the algorithm does not always provide an optimal solution.

Q2-a



alphabet = {A,B,C,D,E}
freuence = {1/3, 1/6, 1/6, 1/6, 1/6}
Q2-b
Define: $\text{fre}(x) = \begin{cases} \text{frequency of } x, & \text{if } x \text{ is a leaf} \\ \textbf{Sum of all frequencies of leaves of the subtree root at x,} & \text{if } x \text{ is an internal node} \end{cases}$

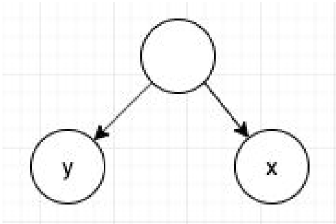
Fact 1: fre(T) = 1 when T is the root of the original tree generated by Huffman's algorithm.
Fact 2: fre(x) > fre(y) => depth(x) <= depth(y)

Proof by contradiction.
Assume ∃ a set of symbols s.t. every symbol has frequency (strictly) less than 1/3 and Huffman's algorithm **can** produce a codeword of length 1.
Let n be the size of the set of symbols.
Suppose x is the symbol with codeword of length 1

Case 1: n = 1
x is the only node in the tree
=> x is the root of the Tree
=> fre(x) = fre(T) = 1 >= 1/3 by fact1

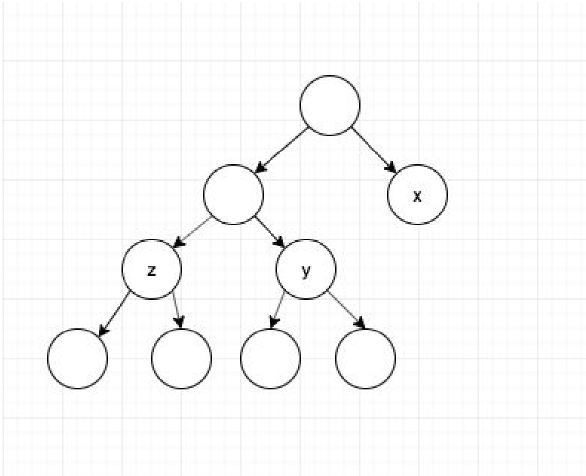
=> contradiction

Case 2: n = 2



=> fre(T) = fre(y) + fre(x) = 1   by fact 1
=> fre(y) = 1 - fre(x)
=> fre(y) > 1 - 1/3   all nodes have frequency less than 1/3 include x (assumption)
=> fre(y) > 2/3 >= 1/3
=> contradiction

Case 3: n >= 3



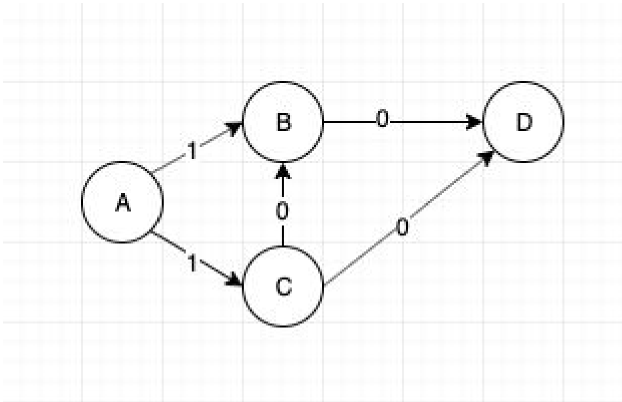
depth(y) > depth(x) and depth(z) > depth(x)
=> fre(y) <= fre(x) and fre(z) <= fre(x)   [by contrapositive of Fact 2]
=> fre(T) = fre(z) + fre(y) + fre(x) <= fre(x) + fre(x) + fre(x)
=> 3fre(x) >= fre(T) = 1
=> fre(x) >= 1/3
=> contradiction

Therefore, our assumption is False, ∀ set of symbols, if every symbol has frequency (strictly) less than 1/3, Huffman's algorithm **cannot** produce a codeword of length 1.

Q3-a
Let n(v) be the number of minimum-weight paths from start node to v
Modified algorithm as following:
R := ∅; d(s) := 0; **n(s) := 1;**
for each v != s do
   d(v) := ∞; **n(v) := 0;**
while R != V do

u := node not in R with min d-value
R := R ∪ {u}
for each v s.t (u,v) ∈ E do
if d(v) > d(u) + wt(u,v) then
d(v) := d(u) + wt(u,v)
n(v) = n(u)
Else if d(v) == d(u) + wt(u,v) then
n(v) = n(v) + n(u)

Q3-b
Above algorithm won't work with zero weight edges!
Below is a counter example:



Let's trace this:

	A	B	C	D
R	-	-	-	-
d	0	∞	∞	∞
n	1	0	0	0

R	+	-	-	-
d	0	1	1	∞
n	1	1	1	0

R	+	+	-	-
d	0	1	1	1
n	1	1	1	1

R	+	+	-	+

d	0	1	1	1
n	1	1	1	1

R	+	+	+	+
d	0	1	1	1
n	1	2	1	2

After the algorithm is finished, we can see n(D) = 2. However there are three minimum-weight paths from A to D: A->B->D, A->C->D and A->C->B->D.