Q1.

(bus, bolt, abacus, aargh, cat)

pivot = bus, i = 0

\( \text{abacus} \text{ aargh} \rightarrow \text{abacus} \rightarrow \text{aargh} \rightarrow s_1 \)  
\( \text{bus} \rightarrow \text{bath} \rightarrow s_2 \)  
\( \text{cat} \rightarrow \text{stop}, \text{out key} \)

Q2.

\( s = (1, 2, 3, 4, 6, 8, 9) \)
\( l = 1 \)
\( low = 1 \)
\( hi = 9 \)

\( m = \frac{hi + low}{2} = 4 \)  
\( s[4] = 4 \)

range \( = [low + m - l, hi + m - r] = [1 + 4 - 1, 9 + 4 - 1] = [4, 6] \)

\( \text{Encode}_2 (4-4) \text{ in } \left\lceil \log_2 6-4+1 \right\rceil = 2 \text{ bits} \rightarrow 00 \)

On the left half:
\( (1, 3, 1, 3) \rightarrow \text{no bit emitted} \)

On the right half:
\( (5, 7, 5, 9) \)
\( l = 5, \text{low}, \text{hi} \)

\( m = 6 \rightarrow s[6] = 8 \)

range \( = [5 + 6 - 5, 9 + 6 - 7] = [6, 8] \)

\( \text{Encode}_2 (8-6) = 2 \text{ in } \left\lceil \log_2 8-6+1 \right\rceil = 2 \rightarrow 10 \)
Q3. \[ P(a) = \frac{1}{2}, \quad P(b) = P(c) = \frac{1}{4} \]

Compressed sequence: \((111)_2 = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} = \frac{7}{8}\)

Q4. \( s_1 = 1, 8 \)

\[ s_2 = 1, 2, 5, 7, 10, 15, 20 \]

\[ s_3 = 8 \]

\[ s_4 = 2, 5, 7, 10, 15, 20 \]

Any search here...

Q5. Given \( P = 110 \) → downward search stops at \( 0 \) and thus any disorderly string can be taken.

Let it be \( s_2 \rightarrow \text{lcp}(P, s_2) = 1 \) and \( s_2 < P \)

Carry upward we stop at the edge \( 0 \) of \( s_2 \) right \( 0 \) → \( P > s_4 \)
For the succinct encoding of a binary tree we proceed as we did in class:

- We do not need to store the first bit of every edge because they are only {0, 1}.
- We need to store the integers of the internal nodes, and we can do so using the BFS order as the succinct encoding.

\[ I = [0, 3, 1, 2, 4, 5, 6, 7] \]

We create the binary sequence that encodes the tree structure:

\[ B = 11100111100000 \]

This is a bijection, mapping between leaf numbers and the entries of \( I \).

We assume to build a Rank/Select data structure on \( B \).

We know that:

- \( B[i] = 1 \) \( \Rightarrow \) node of the complete tree
- \( B[i] = 0 \) \( \Rightarrow \) dummy leaf

\[
\begin{align*}
\text{left child}(\hat{x}) &= \text{rank}_1(2 \cdot \hat{x}) \\
\text{right child}(\hat{x}) &= \text{rank}_1(2 \cdot \hat{x} + 1)
\end{align*}
\]

where \( \hat{x} \) is the number indicated below \( B \) and \( x \) is one of the 1s.

1. We start with \( \hat{x} = 0 \):
   - \( I[\hat{x}] = I[0] = 0 \)

2. \( P[0] = 1 \) \( \Rightarrow \) right child \( (\hat{x}) = \text{rank}_1(2 \cdot 1) \over B = 3 \over 1 \) \( \Rightarrow \)
   - \( B[3] = 1 \)

since \( B[3] = 1 \) the right child does exist, and \( I[3] = 2 \) gives us the value stored in it. Notice that checking the left and right children can discover that it is an internal node.

3. \( P[2] = 0 \) \( \Rightarrow \) left child \( (\hat{x}) = \text{rank}_1(2 \cdot 2) = 4 = \hat{x} \)

4. \( P[4] = 1 \) \( \Rightarrow \) right child \( (\hat{x}) = \text{rank}_1(2 \cdot 4 + 1) \over B = 9 \over 1 \) \( \Rightarrow \)
   - \( B[9] = 1 \)

Since \( B[9] = 1 \) the right child does exist, and \( I[9] = 1 \) gives us the value stored in it. Notice that checking the left and right children can discover that it is an internal node.