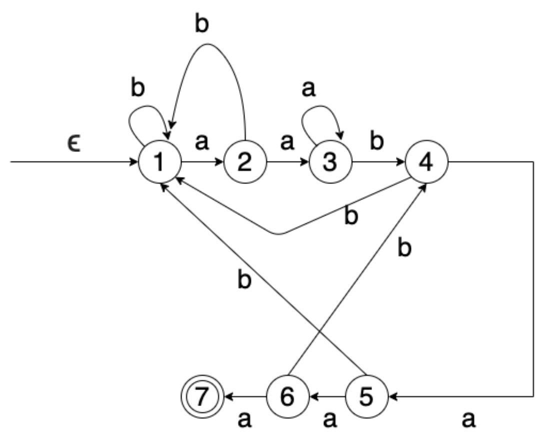
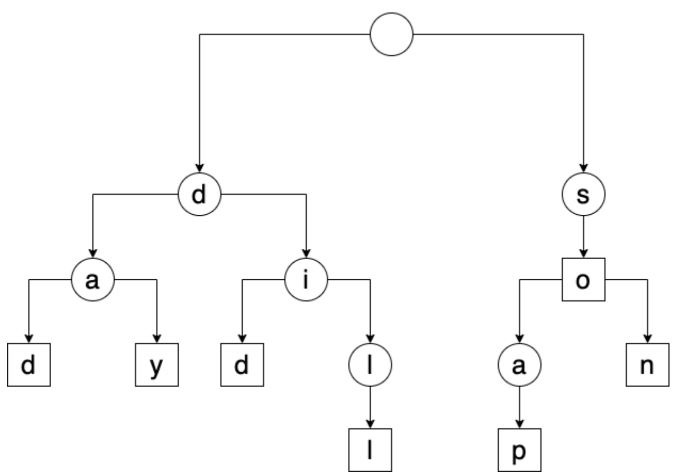
1. 15 points, 5 points per part.

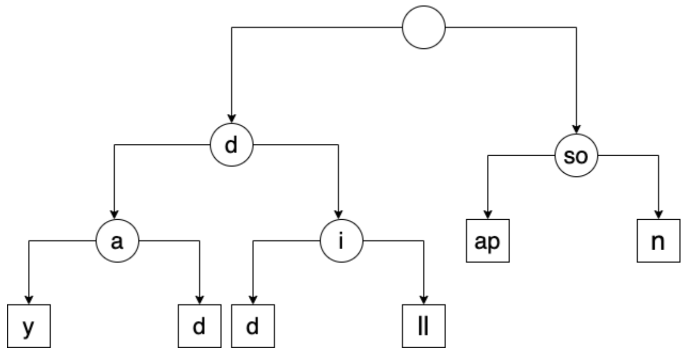
   (a)



   (b)



(c)



2. 5 points. Create a temporary string for example *temp*, and store concatenation of the first string with itself in it, i.e., calculate $S + S$ and store in *temp*. Now, if S' is a sub-string of *temp* then S and S' are rotations of each other.
   Note: The $strstr()$ function finds the first occurrence of the sub-string in the provided string.

   Input: Two strings: S and S'
   Output: $true$ if S' is a rotation of S, $false$ otherwise.

Algorithm 1 Check whether two strings are rotation of one another

$n \leftarrow len(S)$ //store length of S in n
$m \leftarrow len(S'$ //store length of S' in m
if $n\,!= m$ **then**
    $return\ false$
end if
$temp \leftarrow S + S$
if $temp.strstr(S') > 0$ **then**
    $return\ true$
else
    $return\ false$
end if

Time complexity of this problem depends on the implementation of $strstr()$ function. The implementation of $strstr()$ can be done in $\Theta(n + m)$ runtime where n and m are lengths of strings.

3. 20 points, 4 points each.

   (a) True for all $n\%2 = 0$. For all even numbers of n, since its even we can just alternate the two colours.

   (b) True $\forall n > 0$ since a star graph only has the one internal node so 1 colour for that and 1 colour for the external nodes

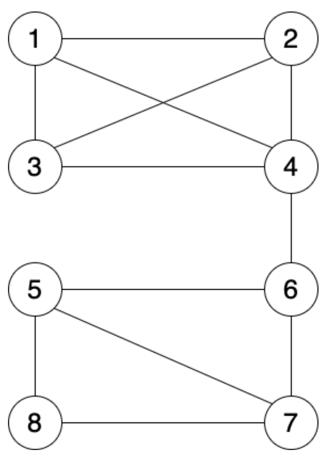
   (c) True for all $n\%2 = 1$. For odd values of n, W_n is a perfect graph with chromatic number 3, the vertices of the cycle can be given 2 colors, and the center vertex given another color.

   (d) Not true for any $n > 0$. All hyper-cubes are bipartite and hence 2-colorable.

   (e) Not true for any $n > 0$. It is a complete bipartite graph and hence 2-colorable.

4. 10 points, 5 points each

   (a)



   (b) I would use adjacency lists to represent a graph with V vertices and E edges if E is $O(V)$ for minimum space requirement, however, if E is $O(V^2)$ we should use adjacency matrix. The adjacency-matrix of any graph has $\Theta(V^2)$ entries, regardless of the number of edges in the graph, so it's better to use it if E is $O(V^2)$. In short, if time is your constraint, use an Adjacency Matrix, if memory is your constraint, use Adjacency List.