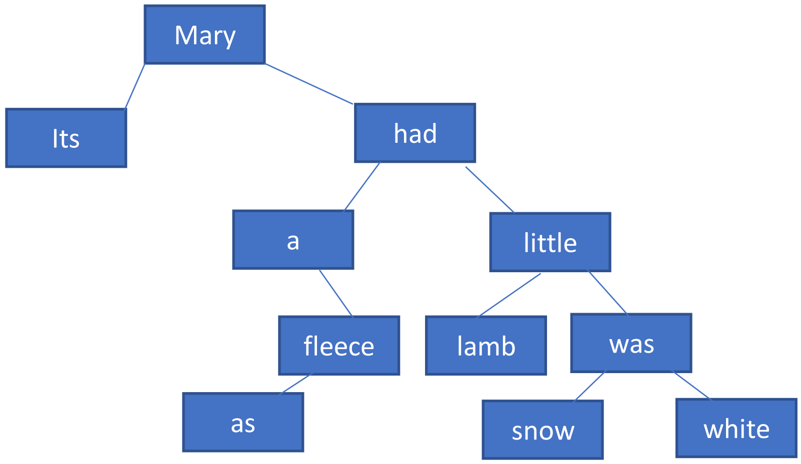
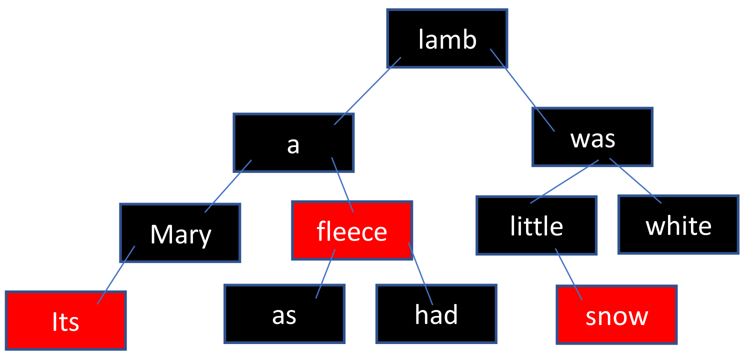
Part 1

1.



2.



3. a. aggregation
   b. inheritance
   c. inheritance
   d. neither

   e. aggregation
   f. inheritance
   g. aggregation
   h. aggregation

4. We can just use a max heap. As we learned in class, building a heap with the amortized way which we first fill every layer one by one and then fix the heap from bottom layer to the top will take O(n) time. As we don't need to delete any flyers in the heap, we can just go through the heap layer by layer and find the log n flyers. With n flyers there are int(log n)+1 layers and we can directly choose the first int(log(log n)) layers' flyers. As O(log n) < O(n) this time can be ignored. The only difference is that when we want to find the last A several flyers, we may have to find these A flyers in the int(log(log n))+1

layer that has more than A flyers. We can just use all the flyers in this layer and build a heap again, which will also take less than O(n) time as there must be less than n flyers in this layer (2^(log(log n)=log n flyers) and can be ignored. Again, we go through layer by layer and find top flyers. Then at the layer with last B flyers left and the flyers in this layer is larger than B, we will do the same thing as last A flyers left, that is build a heap with flyers in this layer and find top flyers. By doing this again and again we will eventually find log n flyers.
The tricky thing here is when we build a new heap with one layer again and again until we find the last flyer, the sum of the time for these operations will become n+log n+log(log n))+…+log(log(…(log n)…)) and its limit will be less than 2n which is also O(n).

Another idea is that we can simply build a heap with amortized way and delete log n times. This will take O((log n)^2) time which is also O(n).