**Question 1:**
**Answer:** We can implement ternary heap just as like the binary heap.

The parent of element $A[i]$ is, $A[floor(i/3)]$.

Advantage of a ternary heap is, it will have $\log_3n$ height. It is $\log_3 2$ order smaller than the binary heap height, $\log_2n$.
Disadvantage is, in heapify/delete_min operation, it will take 2 comparisons to get the minimum of 3 children while swapping for maintaining heap property.

**Question 2:**
**Answer:** We can apply Delete Max operation $k$ times in a Max_Heap to get the top-$k$-largest elements. It will take $O(klogn)$ time. However, using a second heap, we can do this in a faster way. Here goes the description.

Copy the root of the given Max_Heap [lets call it X] and insert it into a new Max_Heap [lets call it Y, this is empty at the beginning]. Apply Delete Max in Y. This will give the first max element.

Copy the first max element's children from X and insert into Y. Apply Delete Max in Y. This will give the second max element.

Copy the second max element's children from X and insert into Y. Apply Delete Max in Y. This will give the third max element.

Do this K times.

Each time, we are inserting two element from X into Y and deleting one element from Y. So, the maximum number of elements in Y is $(k+1)$. So, the height of Y will be $\log k$. As, we are doing $2k$ number of insertion and $k$ number of deletion, the order should be $O(klogk)$.

**Question 3:**
**Answer:** These are the heap operations.

1. Insert  
2. Find min  
3. Delete min.

We can simulate these operations in BST. Below is the description of each of them.

Insert: BST has it's own insert function like Heap. But the disadvantage is, in the worst case, it may take $O(n)$ time unlike $O(logn)$ in Heap.

Find min: In Heap, we can find the min in $O(1)$ time. In BST, the leftmost node contains the minimum number. So, starting from the root, we will need $O(h)$ time to find the min where in the worst case $h = n$ and in the best case $h = logn$. However, there is a better way to find the min in BST. We can keep an extra pointer which will always point to the leftmost node of BST. This pointer will be updated when an insert or delete is made. By this way, we can simulate find min in $O(1)$ time.

Delete min: In Heap, Delete min takes $O(logn)$ time in the worst case. Unlike that, in BST, it will take $O(1)$ time to find the min and $O(h)$ time[in worst case, $h = n$; best case, $h = logn] to change the structure of the tree.
**Question 4:**

**Answer:** BST is implemented using array data structure. We can apply Build_Heap on that array and get the Heap in $O(n)$ time. If it was not implemented using array, we can traverse the BST in $O(n)$ time and get it into an array.