All codes must be justified (either in a separate file or in comments). Your program must be written in C++ (Python is also allowed). Please ONLY send the .h and .cpp files (no executable)!

1) Implement a data structure supporting the following two operations: **/5**
   a. `void init(vector<int> v)`: initialize the data structure content with an n-size vector. Complexity: O(n√n).
   b. `int pow(int i, int j)`: outputs the number of pairs (r,s) s.t. $i \leq r < s \leq j$, $v[r] = 2^{v[s]}$. Complexity: O(√n).

      Some elements of answer: it suffices to apply Mo's trick.

2) We define a ternary tree as follows:

```
struct node {
   int id;
   node *father;
   node *fst, *snd, *thd;
};

typedef node *TernaryTree;
```

   Implement a data structure supporting the following three operations: /5
   a. `void init(TernaryTree T)`: initialize the data structure content with an n-node ternary tree of height O(log(n)). *We assume that all nodes have a unique identifier between 0 and n-1.*
      Complexity: O(n).
   b. `int lca(int i, int j)`: outputs the lowest common ancestor of nodes i,j.
      Complexity: O(1).

      Some elements of answer: it suffices to apply the Binary lifting technique. More precisely, the root is labelled 1, and if a node is labelled i then: i->fst (if it exists) is labelled 4*i, i->snd (if it exists) is labelled 4*i+1, i->thd (if it exists) is labelled 4*i+2.