Please read the following notes before you start:

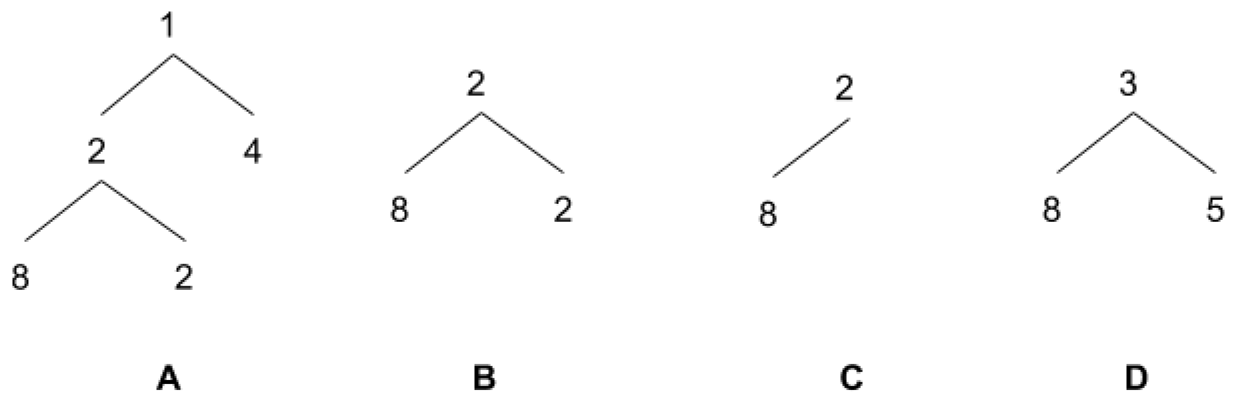
i. You are not required to finish all the questions. Pick the questions that you think can best demonstrate your abilities.
ii. For coding questions, you can use any other languages if you are not familiar with C++.
iii. You can write pseudocode or even just algorithm descriptions if you do not have enough time. However, working code is more appreciated.
iv. State your assumptions clearly.

1) Given a remote server with correct clock time T1, and your machine clock time T2. When your machine is running, T2 may become incorrect. Please design how to use T1 to calibrate T2.

2) Given an array of 10000 integers, with 90% are among 1 to 100, and the others are among 101-10000. How to sort efficiently?

3) Given two binary trees containing integer data, write a function to check if the second tree is a subtree of the first one. Please give a definition for your binary tree first.

For example, in the following diagrams, B is a subtree of A while C and D are not.



4) Given a list of words, sorted in lexicographical order, check if a string is in the list.
For example,  given the list: [“a”, “is”, “this”, “word”], “word” is in the list while “foo” is not.

5) A software package usually has some dependency requirements. For example, if you want to install git, you will need to install libcurl first. Consider the following definition of package:

```
Package {
  string name
  int version
  list<Dependency> deps
}
```

i. Suppose the type Dependency is just a simple pair of (package_name, version). In other words, a dependency is a specific version of a package, e.g. (libcurl, 2).
Write a function that takes in a list containing all packages available, and a list of packages (with the specified versions) the user wants to install, returns the packages the user needs to install, **in the correct order**. The function should report an error if a solution is not possible, e.g. version conflicts, circular dependency, etc.

For example, given the following available packages:
name: A, version: 1, deps: [(B, 1), (C, 2)]
name: A, version: 2, deps: [(B, 1), (C, 3)]
name: B, version: 1, deps: []
name: C, version: 2, deps: []
name: C, version: 3, deps: [(D, 1)]
name: D, version: 1, deps: []

If the user wants to install [(A, 1)], a possible solution is
[(B, 1), (C, 2), (A, 1)]
If the user wants to install [(A, 2)], a possible solution is
[(D, 1), (C, 3), (B, 1), (A, 2)]

ii. If instead of a specific version, the Dependency can also be a range of versions, e.g. "> 1", "< 2", "1 - 3". How would this affect your solution? Describe how to solve this new problem.

6) Please explain the usage of *std::move* and *std::forward*

7) Please rewrite code below without branch conditions/comparisons and justify which one is better. (arr is an integer array)

```
for (int i=0; i != 99999; ++i)
{
        if (arr[i] >= 128) sum+=arr[i];
}
```

8) Define a general tree class (every node can have arbitrary number of children) and an iterator type for this class. Your tree class should support tree traversal via a for-each loop, e.g. `for (auto &node : tree) { … }`. You can choose any order of traversal you like.

9) Roy implements a TCP using UDP, rather than using TCP straightly. Please explain why he does so?

10) Describe some strategies to make an in-memory key-value store, with string keys and string values:

i. Persistent.
ii. Distributed among multiple machines.