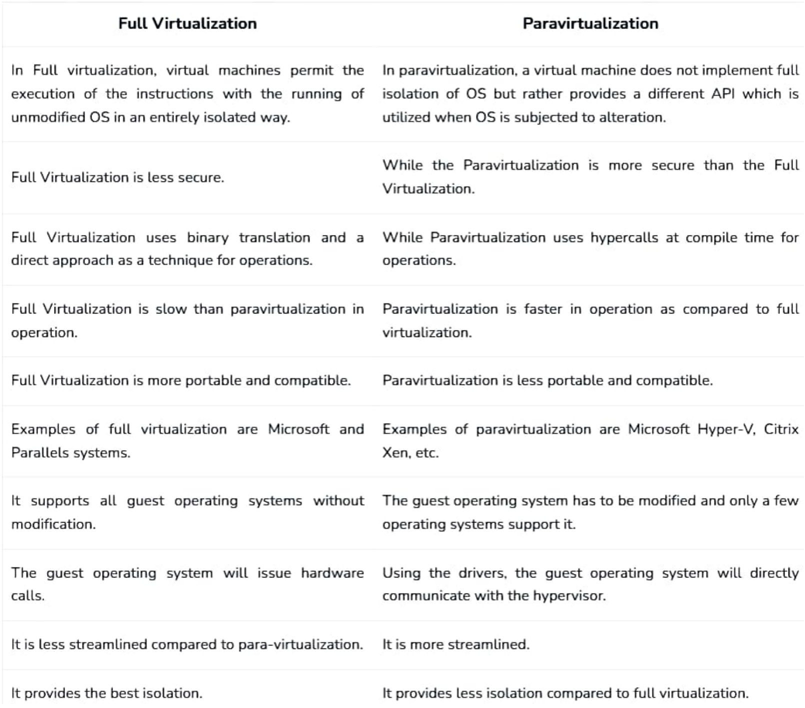
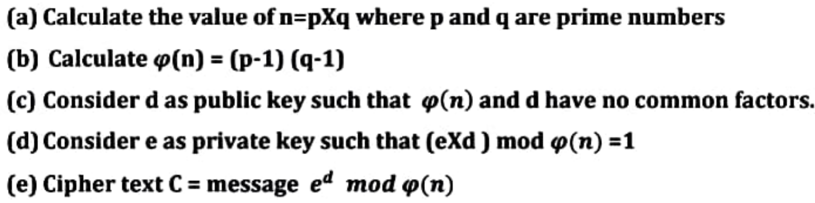


Full Virtualization	Paravirtualization
In Full virtualization, virtual machines permit the execution of the instructions with the running of unmodified OS in an entirely isolated way.	In paravirtualization, a virtual machine does not implement full isolation of OS but rather provides a different API which is utilized when OS is subjected to alteration.
Full Virtualization is less secure.	While the Paravirtualization is more secure than the Full Virtualization.
Full Virtualization uses binary translation and a direct approach as a technique for operations.	While Paravirtualization uses hypercalls at compile time for operations.
Full Virtualization is slow than paravirtualization in operation.	Paravirtualization is faster in operation as compared to full virtualization.
Full Virtualization is more portable and compatible.	Paravirtualization is less portable and compatible.
Examples of full virtualization are Microsoft and Parallels systems.	Examples of paravirtualization are Microsoft Hyper-V, Citrix Xen, etc.
It supports all guest operating systems without modification.	The guest operating system has to be modified and only a few operating systems support it.
The guest operating system will issue hardware calls.	Using the drivers, the guest operating system will directly communicate with the hypervisor.
It is less streamlined compared to para-virtualization.	It is more streamlined.
It provides the best isolation.	It provides less isolation compared to full virtualization.



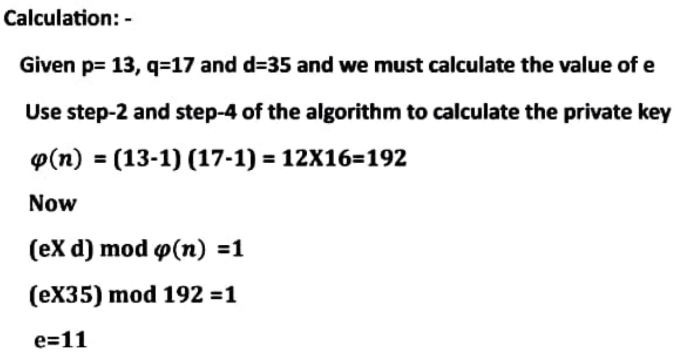
(a) Calculate the value of n=pXq where p and q are prime numbers

(b) Calculate $\varphi(n)$ = (p-1) (q-1)

(c) Consider d as public key such that $\varphi(n)$ and d have no common factors.

(d) Consider e as private key such that (eXd) mod $\varphi(n)$ =1

(e) Cipher text C = message e^d $mod\ \varphi(n)$



Calculation: -

Given p= 13, q=17 and d=35 and we must calculate the value of e

Use step-2 and step-4 of the algorithm to calculate the private key

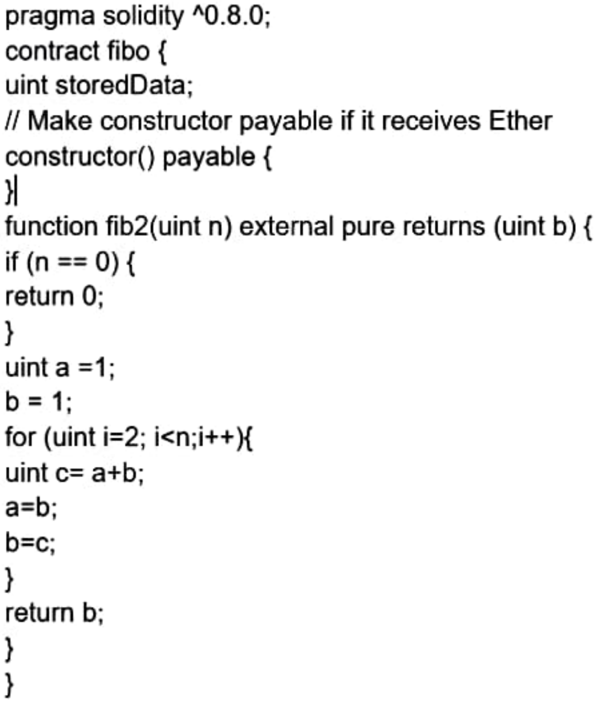
$\varphi(n)$ = (13-1) (17-1) = 12X16=192

Now

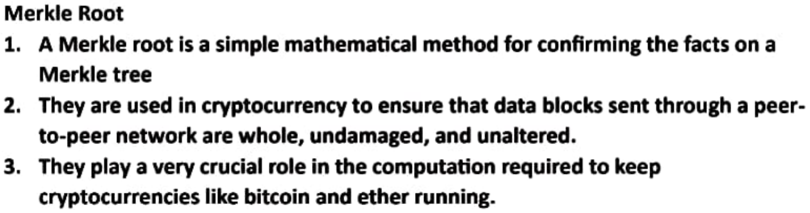
(eX d) mod $\varphi(n)$ =1

(eX35) mod 192 =1

e=11

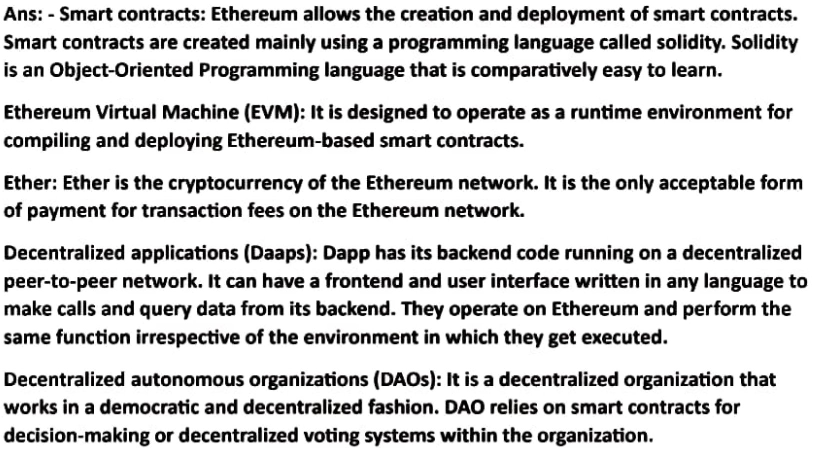


```
pragma solidity ^0.8.0;
contract fibo {
uint storedData;
// Make constructor payable if it receives Ether
constructor() payable {
}
function fib2(uint n) external pure returns (uint b) {
if (n == 0) {
return 0;
}
uint a =1;
b = 1;
for (uint i=2; i<n;i++){
uint c= a+b;
a=b;
b=c;
}
return b;
}
}
```



Merkle Root

1. **A Merkle root is a simple mathematical method for confirming the facts on a Merkle tree**
2. **They are used in cryptocurrency to ensure that data blocks sent through a peer-to-peer network are whole, undamaged, and unaltered.**
3. **They play a very crucial role in the computation required to keep cryptocurrencies like bitcoin and ether running.**



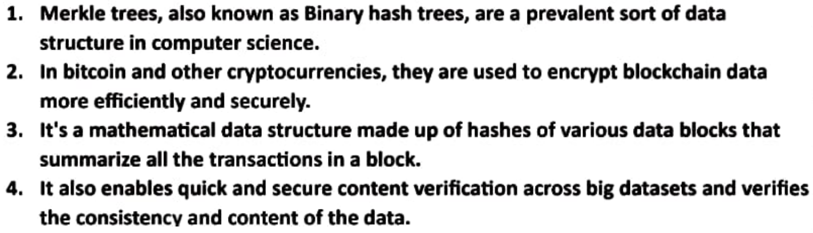
Ans: - Smart contracts: Ethereum allows the creation and deployment of smart contracts. Smart contracts are created mainly using a programming language called solidity. Solidity is an Object-Oriented Programming language that is comparatively easy to learn.

Ethereum Virtual Machine (EVM): It is designed to operate as a runtime environment for compiling and deploying Ethereum-based smart contracts.

Ether: Ether is the cryptocurrency of the Ethereum network. It is the only acceptable form of payment for transaction fees on the Ethereum network.

Decentralized applications (Daaps): Dapp has its backend code running on a decentralized peer-to-peer network. It can have a frontend and user interface written in any language to make calls and query data from its backend. They operate on Ethereum and perform the same function irrespective of the environment in which they get executed.

Decentralized autonomous organizations (DAOs): It is a decentralized organization that works in a democratic and decentralized fashion. DAO relies on smart contracts for decision-making or decentralized voting systems within the organization.



1. **Merkle trees, also known as Binary hash trees, are a prevalent sort of data structure in computer science.**
2. **In bitcoin and other cryptocurrencies, they are used to encrypt blockchain data more efficiently and securely.**
3. **It's a mathematical data structure made up of hashes of various data blocks that summarize all the transactions in a block.**
4. **It also enables quick and secure content verification across big datasets and verifies the consistency and content of the data.**

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.1;
contract Counter {
uint256 private count;
// Constructor to initialize the counter with a
 specific value
constructor (uint256 _initialCount) payable {
count = _initialCount;
}
// Function to increment the counter by 1
function increment() public {
count += 1;
}
// Function to decrement the counter by 1
function decrement () public {
require (count > 0, "Counter: cannot decrement below zero");
count -= 1;
}
// Function to get the current count
function getCount() public view returns (uint256) {
return count;
}
// Function to reset the counter to zero
function reset () public {
count = 0;
}
}
```

For Blockchain Technology details refer to PPT regarding explanation

Hashing in blockchain is a cryptographic process. One where data, like transaction details in a block, is converted into a fixed-length string of characters, known as a hash. This unique digital fingerprint ensures data integrity and immutability. Crucially, even a minor alteration in the original data produces an entirely different hash.

Hashing in blockchain involves using a cryptographic hash function to convert input data into a fixed-size string of characters, known as a hash. This hash uniquely represents the input, making it tamper-resistant.

In blockchain, each block contains a hash of its data, the previous block's hash, and a timestamp. Changing any block's data alters its hash, disrupting the entire chain. This immutability ensures data integrity, enhances security, and facilitates consensus algorithms. Popular blockchain hash functions include SHA-256. The decentralized nature of blockchain relies on hashing and consensus mechanisms to maintain a secure and transparent transaction validation and history.