Information Assurance and Security-IT352

Name:Ankith kumar S H

Roll No:221IT008

Primitive root Algorithm:

```
import readline from "readline";
import { stdin as input, stdout as output } from "process";
const rl = readline.createInterface({ input, output });

// Function to compute GCD
function gcd(a, b) {
  while (b !== 0) {
    [a, b] = [b, a % b];
  }
  return a;
}
// Function to compute (base^exponent) % modulus
function power(base, exponent, modulus) {
  let result = 1;
  base = base % modulus;
  while (exponent > 0) {
    if (exponent % 2 === 1) result = (result * base) % modulus;
```

```
    exponent = Math.floor(exponent / 2);
    base = (base * base) % modulus;
  }
  return result;
}
// Euler's Totient Function φ(n)
function phi(n) {
  let result = n;
  for (let i = 2; i * i <= n; i++) {
    if (n % i === 0) {
      while (n % i === 0) n = Math.floor(n / i);
      result -= Math.floor(result / i);
    }
  }
  if (n > 1) result -= Math.floor(result / n);
  return result;
}
// Function to find the order of r modulo n
function order(r, n) {
  let result = 1;
  let value = r % n;
  while (value !== 1) {
    value = (value * r) % n;
```

```
    result++;
    if (result > n) return -1; // No order
  }
  return result;
}
// Function to find primitive roots of n
function findPrimitiveRoots(n) {
  const primitiveRoots = [];
  const phiN = phi(n);

  for (let r = 2; r < n; r++) {
    if (gcd(r, n) === 1 && order(r, n) === phiN) {
      primitiveRoots.push(r);
    }
  }
  return primitiveRoots;
}
rl.question("Enter a number to find its primitive roots: ", (input) => {
  const n = parseInt(input);
  const primitiveRoots = findPrimitiveRoots(n);
  if (primitiveRoots.length > 0) {
    console.log(`Primitive roots of ${n} are: [${primitiveRoots.join(",
")}]`);
```

```
  } else {
    console.log(`No primitive roots exist for ${n}`);
  }
  rl.close();
});
```

OUTPUT: