Lab 5

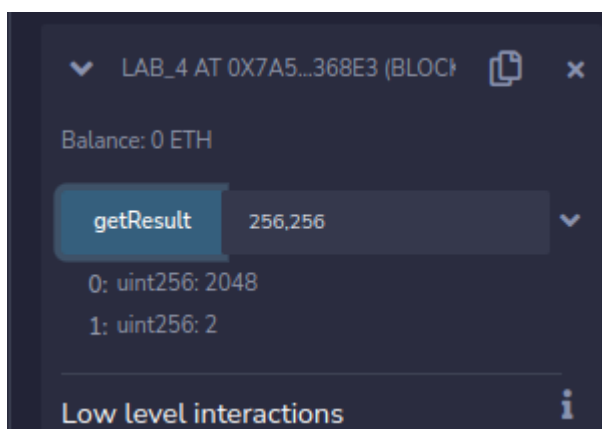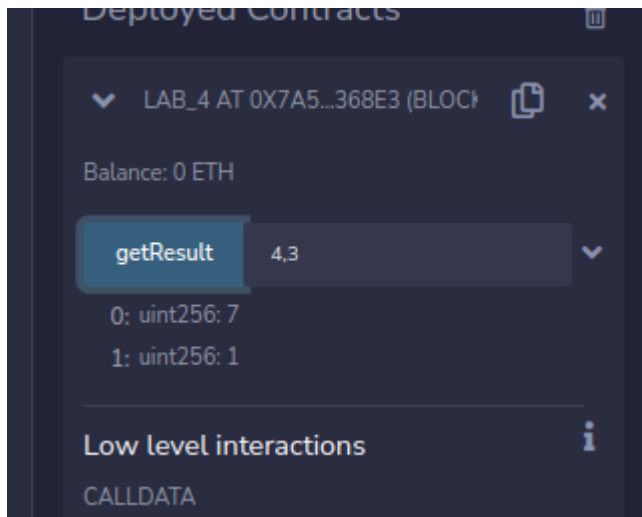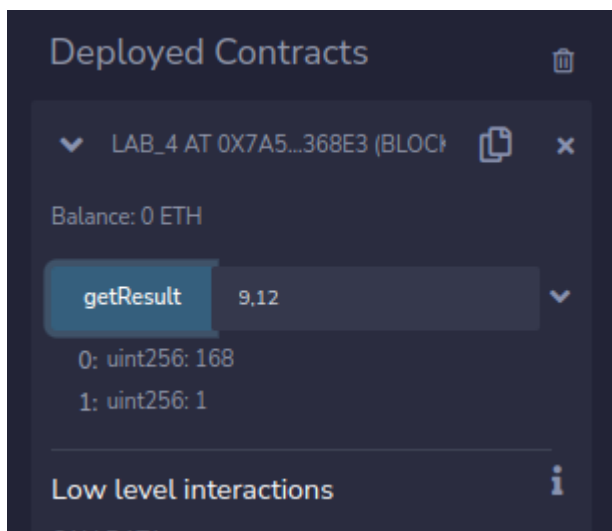## Details :

- Name : P K Navin Shrinivas
- SRN : PES2UG20CS237
- Section : D

Task 1:

- Outputs :

- The outputs follows a logic like so :
  - If the sum of 2 number is a prime number it sum as the first number.
  - Else if multiplies the sum of 2 number with 2, n number of times. Where n is the smallest number sum of 2 numbers (input) is divisible by.
  - The scond number is number of `bytes` requriered to represent the sum of 2 numbers.
- Explaining the loops :
  - Loop 1 : Find the number of bit requried to represent the sum of 2 numbers. It does so by increment the len by 8 if the binary rep range of len is greater than the sum of 2 numbers (_i).
  - Loop 2 : Find the smallest number sum of 2 numbers is divisible by. If not, we set small to 0.
- Replaing `POSISTION3` code :

```
uint len_actual = len/8;
uint j = 2;
uint small =9999;
while (j!=_i){ //LOOP2
    if (_i%j== 0){
        small=j < small? j : small; //POSISTION 3
    }
    j++;
}
if (small==9999){ // POSITION 4
small=0;
}
return (_i<<small,len_actual); // POSITION 5
}
}
```

- Small being assigned to 0 in POSISTION 4 implies that the sum of 2 numbers is a prime number and hence we do not shift the number by anything.
- Returning the 2 numbers without any return statement :

```solidity
pragma solidity ^0.8.7;
contract Lab_4 {
constructor() public{
}

function getResult(uint var1, uint var2) public pure returns(uint, uint ){
uint result = var1 + var2;
uint a;uint b;
Mystery(a,b,result);
return (a,b);
}
function Mystery(uint a,uint b,uint _i) internal {
if (_i == 0) {
a = 0;
b = 1;
}
uint len=8;
while(_i > (2**len)-1) //LOOP 1
{
len+=8;
}
uint len_actual = len/8;
uint j = 2;
uint small =9999;
while (j!=_i){ //LOOP2
    if (_i%j== 0){
        small=j < small? j : small; //POSISTION 3
    }
    j++;
}
if (small==9999){ // POSITION 4
small=0;
}
a = _i<<small;
b = len_actual;
}
}
```

Task 2 :
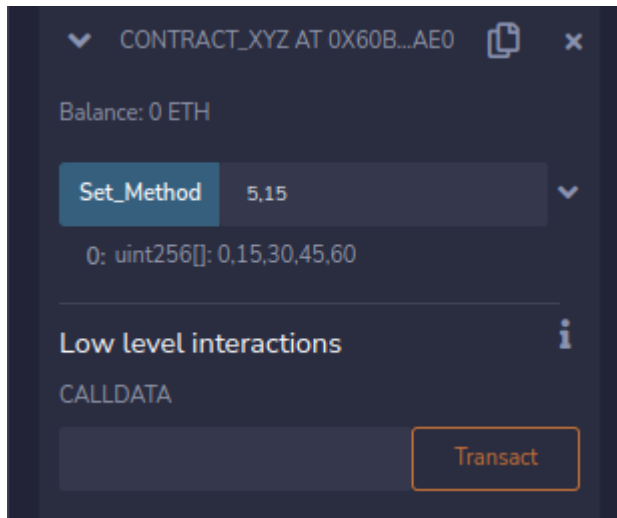
- Code of the contract :

```solidity
pragma solidity ^0.8.7;
contract Contract_XYZ{
        function Set_Method(uint var1, uint var2) public
pure returns(uint[] memory){
                uint[] memory res = new uint[](var1);
                for(uint i=0;i<var1;i++){
                        res[i] = i*var2;
                }
                return res;
```

```
            }
    }
}
```

- Output screenshot :

Task 3:

- Deployment of CODE 1 :

Sepolia test network

Account 1 → New contract

https://remix.ethereum.org

CONTRACT DEPLOYMENT

DETAILS   DATA

🌐 Site suggested › ⓘ

0.00052625

Gas *(estimated)* ⓘ **0.00052625 SepoliaETH**

Very likely in < 15 seconds   **Max fee:**   0.00052625 SepoliaETH

0.00052625

Total   **0.00052625 SepoliaETH**

Amount + gas fee   **Max amount:**   0.00052625 SepoliaETH

Reject   Confirm

- Deployment of CODE 1 :

Sepolia test network

Contract deployment ✕

Status	View on block explorer
Confirmed	Copy transaction ID

From **To**

0xE0F...1833 → New contract

Transaction

Nonce	9
Amount	**-0 SepoliaETH**
Gas Limit (Units)	593191
Gas Used (Units)	593191
Base fee (GWEI)	0.000000011
Priority fee (GWEI)	2.5
Total gas fee	0.001483 SepoliaETH
Max fee per gas	0.000000003 SepoliaETH
Total	**0.00148298 SepoliaETH**

+ Activity log

+ Transaction created with a value of 0 SepoliaETH at 14:40 on 3/30/2023.

Transaction submitted with estimated gas fee of 0.001483 SepoliaETH at 14:40 on 3/30/2023.

✓ Transaction confirmed at 14:41 on 3/30/2023.

+ Transaction data

Assets

Activity

Contract deployment
Mar 30 · remix.ethereum.org
-0 SepoliaETH
-0 SepoliaETH

Contract deployment
Mar 30 · remix.ethereum.org
-0 SepoliaETH
-0 SepoliaETH

Contract deployment
Mar 30 · remix.ethereum.org
-0 SepoliaETH
-0 SepoliaETH

Contract interaction
Mar 30 · remix.ethereum.org
-0 SepoliaETH
-0 SepoliaETH

Contract deployment
Mar 30 · remix.ethereum.org
-0 SepoliaETH
-0 SepoliaETH