// SPDX-License-Identifier: MIT

pragma solidity ^0.8.10;

contract FirstContract

{

    string public greet="Hello World";

}

2.

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.10;

contract Factorial

{

    function facto(uint x) public pure returns(uint)

    {

        uint fact=1;

        uint num;

        uint i;

        num=x;

        for(i=1;i<=x;i++)

        {

            fact= fact\*(i);

        }

        return fact;

    }

}

6.

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.10;

contract CheckDivisibility

{

    function checkDivisibility(uint x) public pure returns(bool)

    {

        uint y=x;

        if(y%7==0 && y%3==0)

        {

            return true;

        }

        else

        {

            return false;

        }

    }

}

7.

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.10;

contract evenOdd

{

    function check(uint num) public pure returns (bool)

    {

        uint x=num;

        uint rem=x % 2;

        if(rem  == 0)

        {

            return true;

        }

        else

        {

            return false;

        }

    }

}

8.

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.10;

contract Calculator

{

    function add(uint a,uint b) public pure returns(uint)

    {

        return (a+b);

    }

    function sub(uint a,uint b) public pure returns(uint)

    {

        return (a-b);

    }

    function mul(uint a,uint b) public pure returns(uint)

    {

        return (a\*b);

    }

    function div(uint a,uint b) public pure returns(uint)

    {

        return (a/b);

    }

}

9.

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.10;

contract SUM

{

    function sumofDigits(uint x) public pure returns(uint)

    {

        uint num;

        uint sum;

        uint rem;

        num=x;

        while(num>0)

        {

           rem=num%10;

           sum=sum+rem;

           num=num/10 ;

        }

        return sum;

    }

}

10.

// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.10;

struct student{

        string Name;

        uint Class;

        string JoiningDate;

    }

contract AttendanceRegister{

    address public Teacher = msg.sender;

    mapping (uint => student) public data;

    event Register(address Teacher, student Data);

    modifier onlyTeacher() {

        require(Teacher == msg.sender, "You are not a teacher");

        \_;

    }

    function add(uint check, string memory name, uint class, string memory date) public onlyTeacher {

        data[check] = student(name,class,date);

        emit Register(msg.sender, data[check]);

    }

}