一、(共58题，共58分)

1.下列编程语言属于静态语言的是\_\_\_\_\_\_。(本题1分)

A）Python

B）Java

C）Javascript

D）PHP

2.下列编程语言属于弱类型语言的是\_\_\_\_\_\_。(本题1分)

A）Python

B）Java

C）Javascript

D）C++

3.截止到2023年9月，Solidity智能合约编程语言的最新版本是\_\_\_\_\_\_。(本题1分)

A）0.7.16;

B）0.8.16;

C）0.8.21;

D）0.8.26;

4.使用\_\_\_\_\_\_预编译指令的Solidity智能合约代码不能被0.8.16版本的编译器编译。(本题1分)

A）pragma solidity ^0.7.16;

B）pragma solidity >=0.7.16;

C）pragma solidity <=0.8.16;

D）pragma solidity <0.8.16;

5.Solidity智能合约代码约定使用\_\_\_\_\_\_进行文档注释。(本题1分)

A）/\*\*…\*/

B）//

C）/\*…\*/

D）///

6.Solidity智能合约ether表示一个以太币，wei表示以太币的最小单位，1ether=\_\_\_\_\_\_wei。(本题1分)

A）1e9

B）1e12

C）1e15

D）1e18

7.enum CompareResult {Invalid, Equal, Less, Greater}定义的枚举数据类型中，Invalid的值为\_\_\_\_\_\_。(本题1分)

A）0

B）1

C）2

D）3

8.调用A合约的addElement函数三次，三次调用给定的参数分别是2、3、4；再给定参数1调用A合约的getElement函数，得到的返回值是\_\_\_\_\_\_。  
contract A {  
    uint256[] private b;  
    function addElement(uint256 number) public {  
        b.push(number);  
    }  
    function getElement(uint256 index) public view returns (uint256) {  
        return b[index];  
    }  
}(本题1分)

A）1

B）2

C）3

D）4

9.给定参数input=0x30313233调用合约A的函数b，则参数input在calldata中实际占用的字节数是\_\_\_\_\_\_。  
contract A {  
    bytes4 c;  
    function b(bytes4 input) public {  
        c = input;  
    }  
}(本题1分)

A）5

B）6

C）32

D）64

10.\_\_\_\_\_\_定义了二维数组（即三个无符号整型不定长数组构成的不定长数组）变量。(本题1分)

A）uint[3][] arrayDemo1;

B）uint[][3] arrayDemo1;

C）int[3][] arrayDemo1;

D）int[][3] arrayDemo1;

11.将CallerContract、CalleeContract这两个合约部署到以太坊中，以CalleeContract合约的地址作为参数调用CallerContract.delegateCallDemo()，返回的结果数据中包含状态变量calledTimes的值，且值是\_\_\_\_\_\_。  
contract CallerContract{  
    uint public calledTimes = 0;  
  
    function delegateCallDemo(address calleeContractAddress) public returns (bool success, bytes memory returnData) {  
        bytes memory payload = abi.encodeWithSignature("functionDemo()");  
        (success, returnData) = calleeContractAddress.delegatecall(payload);  
    }  
}  
  
contract CalleeContract{  
    uint public calledTimes = 10;  
  
    function functionDemo() public returns (uint) {  
        calledTimes++;  
        return calledTimes;  
    }  
}(本题1分)

A）0;

B）1;

C）10;

D）11;

12.合约Demo的函数operation，“return instance.vfunc(input);”这行代码中调用合约DemoA还是合约DemoB的函数vfunc在\_\_\_\_\_\_确定。  
contract DemoA {  
    function vfunc(uint input) virtual public pure returns (uint output) {return input + 2;}  
}  
  
contract DemoB is DemoA{  
    function vfunc(uint inputOne) override public pure returns (uint output) {return inputOne - 2;}  
}  
  
contract Demo {  
    ...  
    function operation(DemoA instance, uint input) internal pure returns (uint output) {return instance.vfunc(input);}  
}(本题1分)

A）合约编译时;

B）合约部署时;

C）合约执行时;

D）合约销毁时;

13.调用合约E的函数c()的返回值是\_\_\_\_\_\_。  
contract C {  
    uint variableC = 30;  
    function c() external view virtual returns(uint) { return 3;}  
}  
  
contract E is C {  
    uint variableE = 50;  
    uint public override c;  
}(本题1分)

A）30;

B）3;

C）50;

D）0;

14.自定义错误类型定义为“error NoPermission(address owner, address caller);”，如果bytes4(keccak256(bytes("NoPermission(address,address)"))的值为0xf6ca7cfa，owner=0x5b38da6a701c568545dcfcb03fcb875f56beddc4，caller=0xab8483f64d9c6d1ecf9b849ae677dd3315835cb2，则捕获该错误类型的lowLevelData是\_\_\_\_\_\_。(本题1分)

A）0xf6ca7cfa5b38da6a701c568545dcfcb03fcb875f56beddc4ab8483f64d9c6d1ecf9b849ae677dd3315835cb2;

B）0xf6ca7cfa5b38da6a701c568545dcfcb03fcb875f56beddc4000000000000000000000000ab8483f64d9c6d1ecf9b849ae677dd3315835cb2000000000000000000000000;

C）0xf6ca7cfa0000000000000000000000005b38da6a701c568545dcfcb03fcb875f56beddc4000000000000000000000000ab8483f64d9c6d1ecf9b849ae677dd3315835cb2;

D）0x000000000000000000000000f6ca7cfa0000000000000000000000005b38da6a701c568545dcfcb03fcb875f56beddc4000000000000000000000000ab8483f64d9c6d1ecf9b849ae677dd3315835cb2;

15.以下合约BytesNDemo中，状态变量bytesVaraible在Storage中对应的存储值是\_\_\_\_\_\_。  
contract BytesNDemo {  
    bytes bytesVaraible;  
  
    constructor () {  
        bytesVaraible = 'abcde';  
    }  
}(本题1分)

A）0xabcde000000000000000000000000000000000000000000000000000005

B）0x6162636465000000000000000000000000000000000000000000000000000005

C）0xabcde00000000000000000000000000000000000000000000000000000a

D）0x616263646500000000000000000000000000000000000000000000000000000a

16.根据错误类型的不同，Solidity支持以下四种异常捕获的方式，捕获所有异常且不关心异常的错误类型和错误信息是\_\_\_\_\_\_。(本题1分)

A）catch Error(string memory reason) { ... }

B）catch Panic(uint errorCode) { ... }

C）catch (bytes memory lowLevelData) { ... }

D）catch { ... }

17.以下哪些数据类型属于值类型\_\_\_\_\_\_。(本题1分)

A）bool

B）function

C）bytes

D）address

18.引用类型的数据存储区域可以显示声明为以下哪些数据存储区域\_\_\_\_\_\_。(本题1分)

A）stack

B）memory

C）storage

D）calldata

19.以下哪些数据类型属于引用类型\_\_\_\_\_\_。(本题1分)

A）mapping

B）struct

C）bytes

D）enum

20.合约Demo的函数notify的参数有哪些数据类型\_\_\_\_\_\_。  
contract Demo {  
    function notify(uint input1, function(bytes memory) external messageCallback, address notifiedAddress) public {  
        ...;  
    }  
}(本题1分)

A）无符号整型

B）不定长字节数组型

C）函数型

D）地址型

21.合约B、C分别继承合约A，合约D继承合约A、B、C，以下正确定义合约D的是\_\_\_\_\_\_。(本题1分)

A）contract D is A, B, C {...}

B）contract D is A, C, B {...}

C）contract D is C, B, A {...}

D）contract D is B, C, A {...}

22.Solidity编程语言是动态语言。(本题1分)

A）对

B）错

23.Solidity编程语言的函数不能定义在contract外。(本题1分)

A）对

B）错

24.合约中定义的状态变量存储在Storage中。(本题1分)

A）对

B）错

25.结构体数据类型是引用数据类型的一种。(本题1分)

A）对

B）错

26.mapping(KeyType => ValueType)用于定义映射数据类型变量，其中KeyType可以使用结构体数据类型。(本题1分)

A）对

B）错

27.函数内定义的bytes2变量，在栈Stack中占用2个字节的空间。(本题1分)

A）对

B）错在Solidity中，所有类型的变量在栈上都占用一个完整的32字节槽位。

28.合约Demo的函数operation中，变量issuerIndex、contryCode的声明和赋值可以成功。

contract Demo {

    function operation() public pure returns(uint8, string memory) {

        uint8 issuerIndex;

        (issuerIndex, string memory contryCode) = (3, "86");

        return (issuerIndex, contryCode);

    }

}(本题1分)

A）对

B）错

29.以下代码中，函数修饰符onlyOwner的require检查在函数transferOwnership之后执行。

contract FunctionModifierDemo {

    address public owner;

    constructor () {

        owner = msg.sender;

    }

    modifier onlyOwner {

        \_;

        require(msg.sender == owner);

    }

    function transferOwnership(address newOwner) public onlyOwner {

        owner = newOwner;

    }

}(本题1分)

A）对

B）错

30.Solidity语言支持重写overwrite。(本题1分)

A）对

B）错

31.合约构造函数中可以使用关键字this。(本题1分)

A）对

B）错

32.可见性为private的函数不能使用关键字virtual声明为可覆盖函数。(本题1分)

A）对

B）错

33.合约Demo中的函数call调用函数operation的方式错误。

contract Demo {

    function operation(uint16 input1, uint16 input2) external pure returns (uint16 output) {

        output = input1 ^ input2;

        return output;

    }

    function call() public pure returns (uint16 output) {

        return operation(1, 2);

    }

}(本题1分)

A）对

B）错

34.将CallerContract、CalleeContract这两个合约部署到以太坊中，以CalleeContract合约的地址作为参数调用CallerContract.staticCallDemo()，返回的结果数据中包含状态变量calledTimes的值，且值是11。

contract CallerContract{

    uint public calledTimes = 0;

    function staticCallDemo(address calleeContractAddress) public view returns (bool success, bytes memory returnData) {

        bytes memory payload = abi.encodeWithSignature("getCalledTimes()");

        (success, returnData) = calleeContractAddress.staticcall(payload);

    }

}

contract CalleeContract{

    uint public calledTimes = 10;

    function functionDemo() public returns (uint) {

        calledTimes++;

        return calledTimes;

    }

}(本题1分)

A）对

B）错

35.合约Demo可以部署在以太坊中。

abstract contract Demo {

    uint varaible;

    constructor(uint param) {

        varaible = param;

   }

}(本题1分)

A）对

B）错

36.assert所创建错误实例的类型为Error(string)(本题1分)

A）对

B）错

37.try/catch语句可以使用以下方式直接捕获自定义错误类型。 catch CustomError(uint errorCode) { ... }(本题1分)

A）对

B）错

38.调用合约Caller的函数operation，该函数将捕获Callee.func()抛出的错误。

contract Callee {

    function func() public pure returns (bool) {

        assert(false);

        return true;

    }

}

contract Caller {

    Callee callee = new Callee();

    function func() public pure returns (bool) {

        revert("intent revert");

    }

    function operation() public view returns (uint result) {

        try callee.func() {

            result = 0;

        } catch Panic(uint errorCode) {

            result = errorCode;

        } catch Error(string memory reason) {

            result = 0xff;

        }

    }

}(本题1分)

A）对

B）错

39.调用合约Caller的函数operation，该函数将捕获Caller.func()抛出的错误。

contract Callee {

    function func() public pure returns (bool) {

        assert(true);

        return true;

    }

}

contract Caller {

    Callee callee = new Callee();

    function func() public pure returns (bool) {

        revert("intent revert");

    }

    function operation() public view returns (uint result) {

        try callee.func() {

            result = 0;

            this.func();

        } catch Panic(uint errorCode) {

            result = errorCode;

        } catch Error(string memory reason) {

            result = 0xff;

        }

    }

}(本题1分)

A）对

B）错

40.给地址支付指定数量以wei为单位的以太币，失败时抛出异常的函数是.transfer(uint256 amount)(本题1分)

A）对

B）错

41.calldata是特殊的数据区域，与memory有类似的行为模式，数据可修改。(本题1分)

A）对

B）错

42.给定参数input1=1024，input2=512调用合约Demo的函数operation，得到的返回值是0x\_\_\_\_\_\_（值中如有字母均小写）。

contract Demo {

    function operation(uint16 input1, uint16 input2) public pure returns (uint16 output) {

        output = input1 ^ input2;

        return output;

    }

}(本题1分)

1）600

43.给定参数input1=1024，input2=1调用合约Demo的函数operation，得到的返回值是0x\_\_\_\_\_\_（值中如有字母均小写）。

contract Demo {

    function operation(uint16 input1, uint16 input2) public pure returns (uint16 output) {

        output = input1 >> input2;

        return output;

    }

}(本题1分)

1）200

44.给定参数0x00112233445566778899aabbccddeeff00112233445566778899aabbccddeeff调用合约AddressDemo的函数addressDemoOne，得到的返回值是0x\_\_\_\_\_\_（值中如有字母均小写）。

contract AddressDemo {

    function addressDemoOne(bytes32 data) public pure returns (address) {

        address addressConverted = address(bytes20(data));

        return addressConverted;

    }

}(本题1分)

1）00112233445566778899aabbccddeeff00112233

45.给定参数0x00112233445566778899aabbccddeeff00112233445566778899aabbccddeeff调用合约AddressDemo的函数addressDemoTwo，得到的返回值是0x\_\_\_\_\_\_（值中如有字母均小写）。

contract AddressDemo {

    function addressDemoTwo(bytes32 data) public pure returns (address) {

        address addressConverted = address(uint160(uint256(data)));

        return addressConverted;

    }

}(本题1分)

1）ccddeeff00112233445566778899aabbccddeeff

46.调用合约A的函数b，得到的返回值是\_\_\_\_\_\_（以十进制表示）。

contract A {

    uint[][4] arrayDemo1;

    constructor () {

        arrayDemo1[0] = [1, 2, 3];

        arrayDemo1[1] = [4, 5, 6, 7];

        arrayDemo1[2] = [8, 9, 10, 11, 12];

        arrayDemo1[3] = [13, 14, 15, 16, 17, 18];

    }

    function b() public view returns (uint){

        return arrayDemo1[2][3];

    }

}(本题1分)

1）11

47.调用合约Demo的函数operation，得到的返回值是\_\_\_\_\_\_（两个返回值之间用中文顿号隔开）。

contract Demo {

    uint[5] arrayVariable = [10,20,30,40,50];

    function operation() public returns (uint, uint){

        assign1(arrayVariable);

        assign2(arrayVariable);

        return (arrayVariable[3], arrayVariable[4]);

    }

    function assign1(uint[5] memory inputArray) internal pure {

        inputArray[3] = 8;

    }

    function assign2(uint[5] storage inputArray) internal {

        inputArray[4] = 10;

    }

}(本题1分)

1）40

2）10

48.调用合约Demo的函数operation，得到的返回值是\_\_\_\_\_\_。

contract Demo {

    function operation() pure public returns (uint) {

        uint variable = 50;

        {

            uint variable;

            variable = 60;

        }

        return variable;

    }

}(本题1分)

1）50

49.type(uint16).max语句执行的十进制结果是\_\_\_\_\_\_。(本题1分)

1）65535

50.面向对象编程OOP的三大特性是\_\_\_\_\_\_。(本题1分)

1）封装

2）继承

3）多态

51.Demo合约operation的返回值十进制表示为\_\_\_\_\_\_。

contract Demo {

    function operation() pure public returns (uint) {

        uint variable = 50;

        {

            uint variable;

            variable = 60;

        }

        return variable >> 2;

    }

}(本题1分)

1）12

52.Demo合约operation函数的第一个返回值十进制表示为\_\_\_\_\_\_。

contract DemoA {

    function operation() pure public returns (uint8, uint8) {

        uint8[3] memory arrayA = [1,2,3];

        uint8[3] memory arrayB = arrayA;

        delete arrayB;

        return (arrayA[2], arrayB[2]);

    }

}(本题1分)

1）3

53.Demo合约operation函数的第二个返回值十进制表示为\_\_\_\_\_\_。

contract DemoA {

    function operation() pure public returns (uint8, uint8) {

        uint8[3] memory arrayA = [1,2,3];

        uint8[3] memory arrayB = arrayA;

        delete arrayB;

        return (arrayA[2], arrayB[2]);

    }

}(本题1分)

1）0

54.ReferenceTypeAssignmentDemo合约referenceTypeAssignmentDemo函数的第一个返回值十进制表示为\_\_\_\_\_\_。

contract ReferenceTypeAssignmentDemo {

    uint[5] arrayVariable = [10,20,30,40,50];

    function referenceTypeAssignmentDemo() public returns (uint, uint){

        memoryArrayAssignment(arrayVariable);

        storageArrayAssignment(arrayVariable);

        return (arrayVariable[3], arrayVariable[4]);

    }

    function memoryArrayAssignment(uint[5] memory inputArray) internal pure {

        inputArray[3] = 8;

    }

    function storageArrayAssignment(uint[5] storage inputArray) internal {

        inputArray[4] = 10;

    }

}(本题1分)

1）40

55.ReferenceTypeAssignmentDemo合约referenceTypeAssignmentDemo函数的第二个返回值十进制表示为\_\_\_\_\_\_。

contract ReferenceTypeAssignmentDemo {

    uint[5] arrayVariable = [10,20,30,40,50];

    function referenceTypeAssignmentDemo() public returns (uint, uint){

        memoryArrayAssignment(arrayVariable);

        storageArrayAssignment(arrayVariable);

        return (arrayVariable[3], arrayVariable[4]);

    }

    function memoryArrayAssignment(uint[5] memory inputArray) internal pure {

        inputArray[3] = 8;

    }

    function storageArrayAssignment(uint[5] storage inputArray) internal {

        inputArray[4] = 10;

    }

}(本题1分)

1）10

56.至少有一个成员函数没有被实现的合约，或者未提供基类合约构造函数参数的合约，或者有意不直接创建的合约是抽象合约，必须用关键字\_\_\_\_\_\_声明。(本题1分)

1）abstract

57.PolymorphismDemo合约函数的输入参数是(2,10,3)，则该函数的返回值十进制表示为\_\_\_\_\_\_。

contract PolymorphismDemoA {

    function vfunc(uint inputOne, uint inputTwo) virtual public pure returns (uint output) {

        return inputOne + inputTwo;

    }

}

contract PolymorphismDemoB is PolymorphismDemoA{

    function vfunc(uint inputOne, uint inputTwo) override public pure returns (uint output) {

        return inputOne - inputTwo;

    }

}

contract PolymorphismDemoC is PolymorphismDemoA {

    function vfunc(uint inputOne, uint inputTwo) override public pure returns (uint output) {

        return inputOne \* inputTwo;

    }}

contract PolymorphismDemoD is PolymorphismDemoA {

    function vfunc(uint inputOne, uint inputTwo) override public pure returns (uint output) {

        return inputOne / inputTwo;

    }

}

contract PolymorphismDemo {

    enum AlgorithmSelector{Add, Reduce, Multiply, Divide}

    PolymorphismDemoA polymorphismDemoA = new PolymorphismDemoA();

    PolymorphismDemoB polymorphismDemoB = new PolymorphismDemoB();

    PolymorphismDemoC polymorphismDemoC = new PolymorphismDemoC();

    PolymorphismDemoD polymorphismDemoD = new PolymorphismDemoD();

    function polymorphismDemo(AlgorithmSelector algorithmSelector, uint inputOne, uint inputTwo)     public view     returns (uint output) {

        if (algorithmSelector == AlgorithmSelector.Add) {

            output = polymorphismCall(polymorphismDemoA, inputOne, inputTwo);

        }

        else if (algorithmSelector == AlgorithmSelector.Reduce) {

            output = polymorphismCall(polymorphismDemoB, inputOne, inputTwo);

        }

        else if (algorithmSelector == AlgorithmSelector.Multiply) {

            output = polymorphismCall(polymorphismDemoC, inputOne, inputTwo);

        }

        else if (algorithmSelector == AlgorithmSelector.Divide) {

            output = polymorphismCall(polymorphismDemoD, inputOne, inputTwo);

        }

    }

    function polymorphismCall(PolymorphismDemoA instance, uint inputOne, uint inputTwo) internal pure returns (uint output) {

        // it is determined which function is called only when below code line being executed.

        return instance.vfunc(inputOne, inputTwo);

    }

}(本题1分)

1）30

58.assert函数和require函数用于判断条件是否满足，并在条件不满足时抛出异常（exception）。返回剩余燃料的是\_\_\_\_\_\_函数。(本题1分)

1）require