Default Question Block

Thank you for participating in our experiment! You have the opportunity to help shape the future of programming language design, enabling safer and easier development of high-stakes software. The study will take about an hour to complete, but you will be given opportunities for breaks in the middle. Once you start the study, please finish it within 24 hours.

You will now be asked to answer some programming questions. We are interested both in your answers and in how long you take to answer the questions. If you need a break, please take one now; afterward, please focus on the questions without getting distracted until the survey is done.

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Read the document "Ownership -- Introduction". Once you are finished, answer the following questions. You may refer to the document as needed.

These page timer metrics will not be displayed to the recipient.

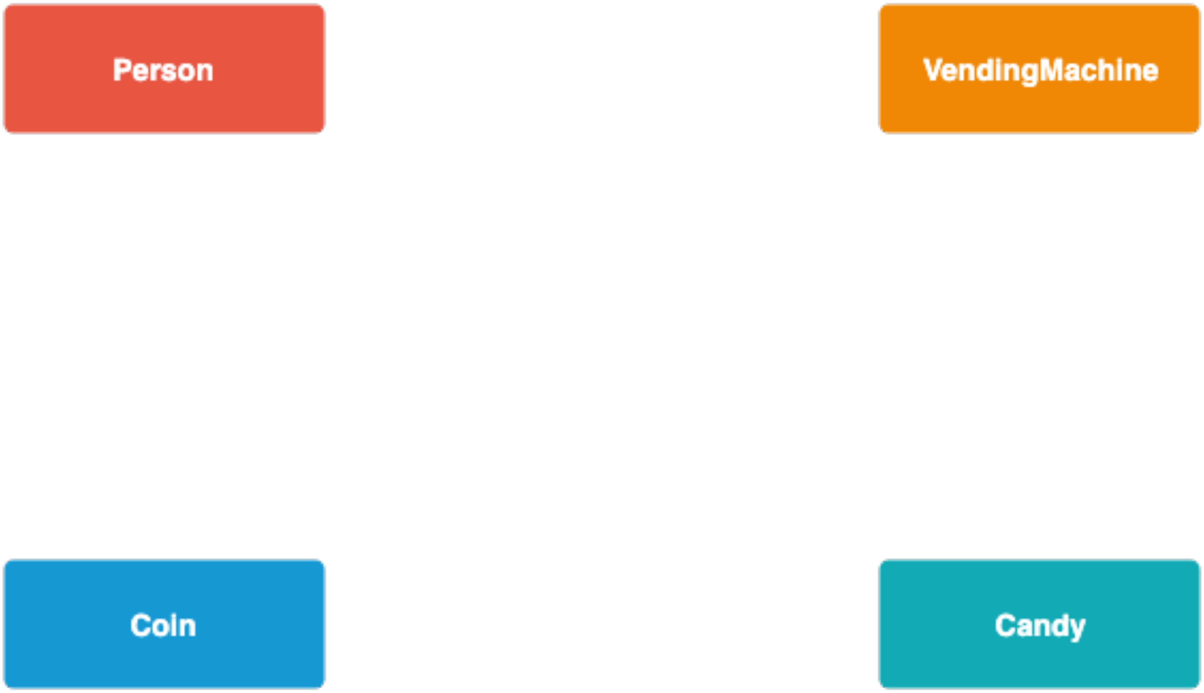
First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

The following two questions refer to this diagram. A `Person` has a `Coin` which they want to use at a `VendingMachine` to acquire a `Candy`. When the `Person` places a `Coin` in the `VendingMachine`, they get a `Candy` in return. You may assume that initially, a `Person` doesn't have any `Candy`, and a `VendingMachine` doesn't have any `Coins`.



Write the ownership references that are true BEFORE the Person places the Coin in the VendingMachine, based on the objects shown in the diagram above. One reference is given to you as an example for formatting (note it is NOT accurate); please follow this format for your references.

Example: Candy --> Coin // Owned (This means Candy has an Owned reference to a Coin object)

Write the ownership references that are true AFTER the Person places the Coin in the VendingMachine, based on the objects shown in the diagram above.

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Write a contract called `Person` that has an `Owned` reference to a `House` and a `Shared` reference to a `Park`. The `House` and `Park` contracts are given below.

```
contract House {

}

contract Park {

}
```

Please write your answer in the VSCode window (code1.sol). You may compile your code to check your answer.

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Read the document "Ownership -- Functions". Once you are finished, answer the following questions. You may refer to the document as needed (and any previous ones).

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Refer to the code below for the next five questions.

```
contract Money {
    ...
}

contract Wallet {
    Money m; // m is owned

    constructor() public {
        m = new Money();
    }

    // returns owned Money
    function spendMoney() public returns (Money) {
        ...
    }

    // mon is initially owned but must be unowned afterward.
    function receiveMoney(Money mon) public {
        ...
    }
}
```

What is **m** in the code fragment above?

- ○ A Money object
- ○ An owned reference to a Money object
- ○ An Owned object
- ○ All of the above
- ○ None of the above

What is the return type of the spendMoney function (if one exists)?

What is the return type of the receiveMoney function (if one exists)?

Assuming correct implementation, what is the Ownership status of `mon` at the **beginning** of the receiveMoney function?

Assuming correct implementation, what is the Ownership status of `mon` at the **end** of the receiveMoney function?

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Refer to the following code for the next two questions.

```
contract Share {

}

contract ShareHolder {
    Share share; // share is owned

    constructor() public {
        share = new Share();
    }

    function receiveShare( Share s) public { // TODO
        ...
    }

    // s is owned
    function checkShareValue(Share s) public {
        ...
    }
}
```

What comment goes in the // TODO above?
(note: s is a Share owned by another ShareHolder that is passed as a parameter to the receiveShare function, and will be set to the field "share")

Please write your answer in the VSCode window (code2.sol). You may compile your code to check your answer.

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Refer to the following code for the next two questions.

```
contract Share {

}

contract ShareHolder {
    Share share; // share is owned

    constructor() public {
        share = new Share();
    }

    function receiveShare( Share s) public { // TODO
        ...
    }
    // s is owned
    function checkShareValue(Share s) public {
        ...
    }
}
```

What is an equivalent way to comment about s in the checkShareValue function declaration?

○ // s is shared
○ // s is unowned intially but owned afterward
○ // s is owned intially and owned afterward
○ // s is unowned intially and unowned afterward
○ None of the above

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Read the document "Ownership -- Variables". Once you are finished, answer the following questions. You may refer to the document as needed (and any previous ones).

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Refer to the following code for the next three questions.

```
contract Money {
    int public amount;
}

contract Wallet {
    Money m; // m is owned

    constructor() public {
        m = new Money();
    }

    function spendMoney() public {
        //...
    }

    // mon is owned at the beginning but unowned at the end
    // Returns an owned reference.
    function receiveMoney(Money mon) public returns (Money) {
        Money temp = m;
        m = mon;
        return temp;
    }

    // Returns an owned reference.
    function checkMoney() public returns (Money) {
        return m;
    }

}
```

What must be the ownership status of m after every function in Wallet?

What occurs in the receiveMoney function with regards to ownership? Is m's ownership status before the function begins the same as when it ends?

Is checkMoney's implementation consistent with its declaration, comment about ownership, and the surrounding code?

- ○ No, you cannot return a field of a contract in a function
- ○ Yes, the return type matches the type of m
- ○ No, returning m makes it unowned, which doesn't match the ownership status of m's declaration
- ○ Yes, m is of type owned, which matches the ownership status of m's declaration
- ○ None of the above

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Write a constructor for the `Money` object so that it accepts an integer as a parameter and sets "amount" to that integer value.
Note: you will have to change the instantiation of m in Wallet for your code to compile.
Please write your answer in the VSCode window (code3.sol). You may compile your code to check your answer.

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Read the document "Ownership -- Miscellaneous". Once you are finished, answer the following questions. You may refer to the document as needed (and any previous ones).

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Definitions of Money and Wallet contracts for the following questions.

```
contract Money {
    int amount;

    function getAmount() public returns (int) {
        return amount;
    }
}

contract Wallet {
    // m is owned
    Money m;

    constructor() public {
        m = new Money();
    }

    function spendMoney() public {
        ...
    }

    // mon is owned initially but unowned at the end.
    // Returns an owned reference.
    function replaceMoney(Money mon) public returns (Money) {
        Money temp = m;
        m = mon;
        return temp;
    }
}
```

Refer to the following code for the next question. Assume the definitions of Wallet and Money are as shown above.

```
// w and m are both owned
function foo (Wallet w, Money m) {

    w.replaceMoney(m);
    w.spendMoney();
    w.replaceMoney(m);

}
```

What is the problem with the function "foo"? Give your answer in terms of ownership.

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Write a Person contract that has an owned reference to a Wallet object. Person should also have an addMoney() function that takes in a Money parameter and passes it to its Wallet's replaceMoney() function. addMoney() should return the Wallet's initial money that was replaced.
Please write your answer in the VSCode window (code4.sol). You may compile your code to check your answer.

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Read the document "Assets". Once you are finished, answer the following questions. You may refer to the document as needed (and any previous ones).

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Refer to the following code for the next two questions.

```
// Medicine is an asset
contract Medicine {

}

// Pharmacy is an asset
contract Pharmacy {
    // med is owned
    Medicine med;

    // Initially, m is owned; afterward, m is unowned.
    function getNewMedicine(Medicine m) public {
        med = m;
    }

    function throwAwayMedicine() public {
        // disown med
    }
}
```

What is the problem (if one exists) with the getNewMedicine function?

○ med becomes Unowned, although it should be Owned at the end of the function
○ m is stated as becoming an Unowned reference, but actually stays Owned
○ The owning reference to m is lost
○ The owning reference to the original Medicine object (med) is lost
○ There is no error

What is the problem (if one exists) with the throwAwayMedicine function?

○ An Owned reference cannot be disowned
○ med is Unowned in the function, so it cannot be disowned
○ med becomes Unowned after this function; this doesn't match its declaration
○ Only a field can be disowned
○ There is no error

How could you fix getNewMedicine? Describe your answer in words/pseudocode. Write N/A if there is no problem with the function.

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Read the document "States -- Introduction". Once you are finished, answer the following questions. You may refer to the document as needed (and any previous ones).

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

A `Drink` can be hot, cold, or lukewarm. If the `Drink` is cold, it has an `IceCube` object. If it is hot, it has a `CupSleeve`. The `Drink` always has an integer value to keep track of its temperature, and always starts lukewarm.

Write the `Drink` contract with the necessary states and fields as described above. NOTE: You do NOT need to write a constructor for `Drink` (you can ignore this compiler error for now)

Please write your answer in the VSCode window (code5.sol). You may compile your code to check your answer.

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Describe the relationship between states and ownership in your own words.

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Read the document "States -- Manipulating State". Once you are finished, answer the following questions. You may refer to the document as needed (and any previous ones).

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Refer to the following code for the next four questions.

```
contract Voter {
    string name;
    bool citizen;
    enum State { Eligible, Ineligible, Registered, FinishedVoting }
    State state;

    constructor(string n, bool citizenship) public {
        name = n;
        citizen = citizenship;
        if (!citizen) {
            state = State.Ineligible;
        }
        else {
            state = State.Eligible;
        }
    }

    // Transitions the voter from Registered to FinishedVoting.
    function vote() public {
        // TODO
    }

    // Transitions the voter from Eligible to either Registered or Inel
    function register() public {
        ...
    }
}
```

What state(s) will a Voter be in after it is first instantiated?

What state(s) will a Voter be in after the function "register"?

What state(s) must a Voter be in to vote?

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Finish the `vote` function by transitioning the state of a `Voter` to the required state. Please write your answer in the VSCode window (code6.sol). You may compile your code to check your answer.

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Refer to the following code for the next two questions.

```
contract Candy {
}

contract Coin {
}

contract CoinBag {
    // Takes ownership of c.
    function deposit(Coin c) public {
        // Real implementation not shown; for now, just throw c away.
        // disown c;
    }
}

contract TinyVendingMachine {
    CoinBag coinBin;
    Candy inventory; // only when in Full state

    enum State {Full, Empty}
    State state;

    // Starts Empty.
    constructor () public {
        coinBin = new CoinBag();
        state = State.Empty;
    }

    // Transitions the receiver from Empty to Full.
    // Consumes ownership of c.
    function restock(Candy c) public {
        require(state == State.Empty, "Can only restock in Empty state.
        // TODO
    }

    // Returns Candy owned by the caller.
    function buy(Coin c) public returns (Candy) {
        require(state == State.Full, "Can only buy in Full state.");
        coinBin.deposit(c);
        Candy result = inventory;
        state = State.Empty;
        return result;
    }
}
```

Write code to complete the `buy` function declaration above.
Please write your answer in the VSCode window (code7.sol). You may compile your code to check your answer.

Complete the `restock` function above by transitioning the state of `TinyVendingMachine` to `Full`.
Please write your answer in the VSCode window (code7.sol). You may compile your code to check your answer.

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Read the documents "States -- Miscellaneous" and "States and Assets". Once you are finished, answer the following questions. You may refer to the document as needed (and any previous ones).

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

Which of the following can be passed as a parameter to the following function (assume `Money` is an **asset**):

```
// money is owned
function foo(Money money) {

}
```

You may choose more than one answer.

☐ Money m; // m is owned
☐ Money m; // m is unowned
☐ Money m; // m is shared
☐ None of the above

Which of the following can be passed as a parameter to the following function (assume `Money` is an **asset**):

```
// money is unowned
function foo(Money money) public {

}
```

You may choose more than one answer.

☐ Money m; // m is owned

☐ Money m; // m is unowned
☐ Money m; // m is shared
☐ None of the above

Which of the following can be passed as a parameter to the following function (assume `Money` is an **asset**):

```
// money is shared
function foo(Money money) public {
}
```

You may choose more than one answer.

☐ Money m; // m is owned
☐ Money m; // m is unowned
☐ Money m; // m is shared
☐ None of the above

Read the document "Using Solidity on a Blockchain." Once you are finished, answer the following questions. You may refer to the document as needed (and any previous ones).

Consider:

```
contract KangarooTracker {
    int public kangaroos;

    function f() {
       // Need to access 'kangaroos' field
    }
}
```

Which of the following will access the contents of the kangaroos field from f?

○ kangaroos()
○ getKangaroos()

○ None of the above; only accessing 'kangaroos' directly works since no getter was defined above.

```
contract LightSwitch {
    enum State { Off, On }
    State state;
}
```

Which of the following is a valid assignment to the state field?

○ state = State(Off);

○ state = State.Off;

○ state = Off;