Name: Prashant

Roll No: 24B2158

Week 1: HelloWorld Contract

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract HelloWorld {
    string public message;

    constructor() {
        message = "Hello, World!";
    }

    function setMessage(string calldata newMessage) external {
        message = newMessage;
    }
}
```

Week 2: Struct and State Variables

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Crowdfunding {
    struct Campaign {
        address owner;
        uint goal;
        uint deadline;
        string title;
        string description;
        uint fundsRaised;
        bool isOpen;
    }

    uint public campaignCount = 0;
    mapping(uint => Campaign) public campaigns;
```

2.1: Create Campaign Function

```
    function createCampaign(
        uint _goal,
        uint _duration,
        string memory _title,
        string memory _description
    ) public {
        campaigns[campaignCount] = Campaign(
            msg.sender,
            _goal,
            block.timestamp + _duration,
            _title,
            _description,
            0,
            true
        );
        campaignCount++;
    }
}
```

Week 3: Contribution Mapping & Setup

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Crowdfunding {
    struct Campaign {
        address owner;
        uint goal;
        uint deadline;
        string title;
        string description;
        uint fundsRaised;
        bool isOpen;
    }

    uint public campaignCount = 0;
    mapping(uint => Campaign) public campaigns;
    mapping(uint => mapping(address => uint)) public contributions;
```

3.1: Create Campaign Function

```
    function createCampaign(
        uint _goal,
        uint _duration,
        string memory _title,
        string memory _description
    ) public {
        campaigns[campaignCount] = Campaign(
            msg.sender,
            _goal,
            block.timestamp + _duration,
            _title,
            _description,
            0,
            true
        );
        campaignCount++;
    }
```

3.2: Contribute Ether Function

```
    function contribute(uint _campaignId) public payable {
        Campaign storage campaign = campaigns[_campaignId];
        require(campaign.isOpen, "Campaign closed");
        require(block.timestamp < campaign.deadline, "Deadline
passed");
        require(msg.value > 0, "Must send some Ether");

        campaign.fundsRaised += msg.value;
        contributions[_campaignId][msg.sender] += msg.value;
    }
}
```

Week 4: Extended Struct & Modifiers

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Crowdfunding {
    struct Campaign {
        address owner;
        uint goal;
        uint deadline;
        string title;
        string description;
        uint fundsRaised;
        bool isOpen;
        bool fundsWithdrawn;
    }

    uint public campaignCount = 0;
    mapping(uint => Campaign) public campaigns;
    mapping(uint => mapping(address => uint)) public contributions;

    modifier onlyOwner(uint _campaignId) {
        require(msg.sender == campaigns[_campaignId].owner, "Not
owner");
        _;
    }
```

4.1: Create Campaign Function

```
    function createCampaign(
        uint _goal,
        uint _duration,
        string memory _title,
        string memory _description
    ) public {
        campaigns[campaignCount] = Campaign(
            msg.sender,
            _goal,
            block.timestamp + _duration,
            _title,
            _description,
            0,
            true,
            false
        );
        campaignCount++;
    }
```

4.2: Contribute Ether Function

```
    function contribute(uint _campaignId) public payable {
        Campaign storage campaign = campaigns[_campaignId];
        require(campaign.isOpen, "Campaign closed");
        require(block.timestamp < campaign.deadline, "Deadline
passed");
        require(msg.value > 0, "Must send Ether");

        campaign.fundsRaised += msg.value;
        contributions[_campaignId][msg.sender] += msg.value;
    }
```

4.3: Withdraw Funds Function

```
    function withdrawFunds(uint _campaignId) public
onlyOwner(_campaignId) {
        Campaign storage campaign = campaigns[_campaignId];
        require(block.timestamp >= campaign.deadline, "Deadline not
reached");
        require(campaign.fundsRaised >= campaign.goal, "Goal not
reached");
        require(!campaign.fundsWithdrawn, "Already withdrawn");

        campaign.fundsWithdrawn = true;
        campaign.isOpen = false;
        payable(campaign.owner).transfer(campaign.fundsRaised);
    }
```

4.5: Refund Contributors Function

```
    function refund(uint _campaignId) public {
        Campaign storage campaign = campaigns[_campaignId];
        require(block.timestamp >= campaign.deadline, "Deadline not 
reached");
        require(campaign.fundsRaised < campaign.goal, "Goal was 
reached");

        uint amount = contributions[_campaignId][msg.sender];
        require(amount > 0, "No contributions");

        contributions[_campaignId][msg.sender] = 0;
        payable(msg.sender).transfer(amount);
    }
}
```