Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Лабораторна робота №1

Програмування смарт-контрактів

Тема: Типи даних в мові Solidity

Виконав
студент групи ІП-11:
Панченко С. В.

Перевірив:
Яланецький В.А.

Київ 2025

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Ознайомитися з середовищем Remix та базовими поняттями про смарт-контракти.

2 ЗАВДАННЯ

Створити смарт-контракт в Remix, додати до нього необхідні параметри та функції згідно індивідуального варіанту, протестувати

його роботу.

1. Засобами IDE створити смарт-контракт із назвою [SN]. Створити бібліотеку із назвою [PIP]. Наповнити смарт-контракт та бібліотеку потрібними змінними та методами, в назвах яких використовувати префікс [PIP].
2. Смарт-контракт отримує [DD] ETH від кожної з [MM]/2 адрес (не менше 3).
3. Смарт-контракт приймає текстовий рядок із ринковим курсом [YYYY], наприклад, отримує «ETH=3200$» та повертає 3200.
4. Смарт-контракт приймає числову температуру повітря [MM] та повертає текстовий коментар про погоду із вказання температури, наприклад, отримує -2 та повертає «холодно: -2 C».

3 ВИКОНАННЯ

3.1 PanchenkoSerhiiVitaliyovichContribute

Для початку на рисунку 3.1 задеплоїмо контракт.

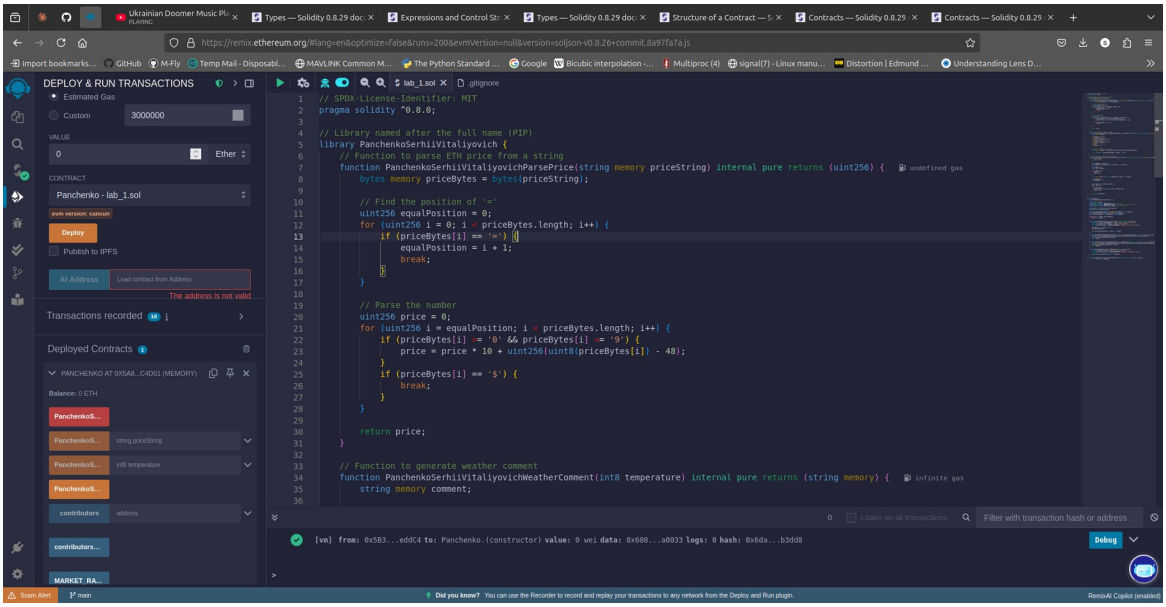
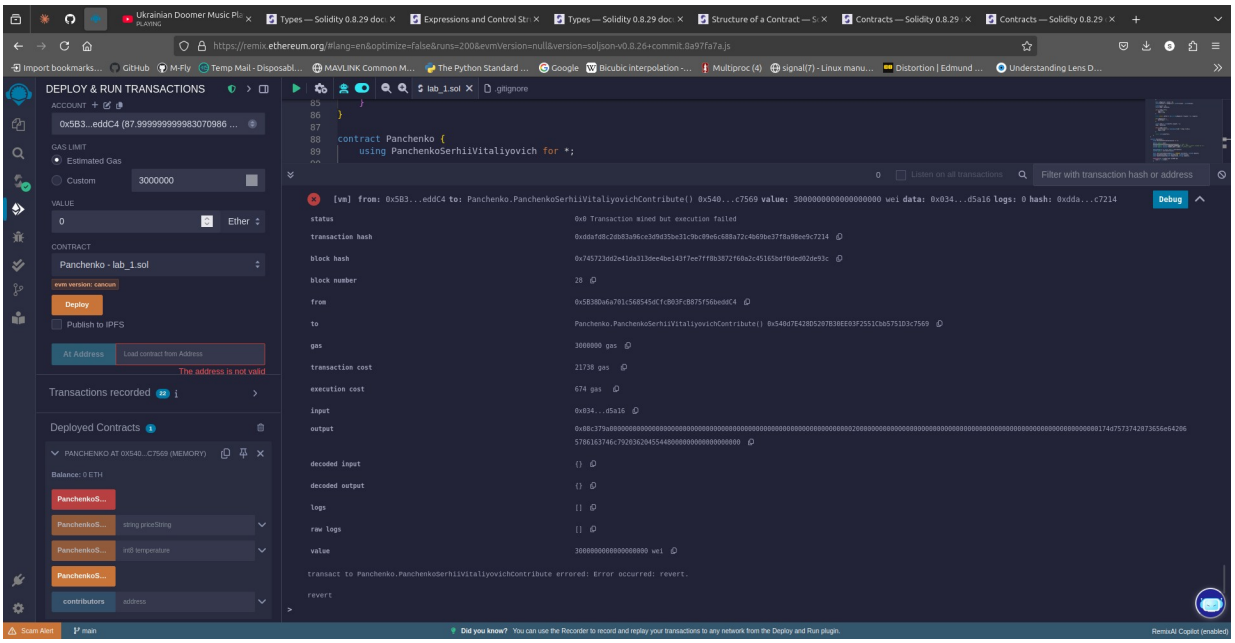


Рисунок 3.1 — задеплоїний контракт

Оскільки мій день народження 6-го березня, то необхідна кількість ефіру має бути 6. На рисунку 3.2 бачимо, що при надсиланні ефіру не в кількості 6-ти одиниць викидує помилку.



Рисунко 3.2 — контракт видає помилку при неправильній кількості надісланих ETH

[vm]

from: 0x5B3...eddC4

to: Panchenko.PanchenkoSerhiiVitaliyovichContribute() 0x540...c7569

value: 3000000000000000000 wei

data: 0x034...d5a16

logs: 0

hash: 0xdda...c7214

status 0x0 Transaction mined but execution failed

transaction hash

0xddafd8c2db83a96ce3d9d35be31c9bc09e6c688a72c4b69be37f8a98ee9c7214

block hash

0x745723dd2e41da313dee4be143f7ee7ff8b3872f60a2c45165bdf0ded02de93c

block number 28

from 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

to Panchenko.PanchenkoSerhiiVitaliyovichContribute()

0x540d7E428D5207B30EE03F2551Cbb5751D3c7569

gas 3000000 gas

transaction cost 21738 gas

execution cost 674 gas

input 0x034...d5a16

output

0x08c379a0000000000000000000000000000000000000000000000000000000000000002000000000000000000000000000000000000000000000000000000000000001746d7573742073656e642065786163746c792036204554480000000000000000

decoded input {}

decoded output {}

logs []

raw logs []

value 3000000000000000000 wei

transact to Panchenko.PanchenkoSerhiiVitaliyovichContribute errored: Error occurred: revert.

revert

The transaction has been reverted to the initial state.

Reason provided by the contract: "Must send exactly 6 ETH".

If the transaction failed for not having enough gas, try increasing the gas limit gently.

transact to Panchenko.PanchenkoSerhiiVitaliyovichContribute pending ...

На рисунку бачимо, що при надсиланні ефіру в кількості 6-ти одиниць контракт приймає повідомлення.

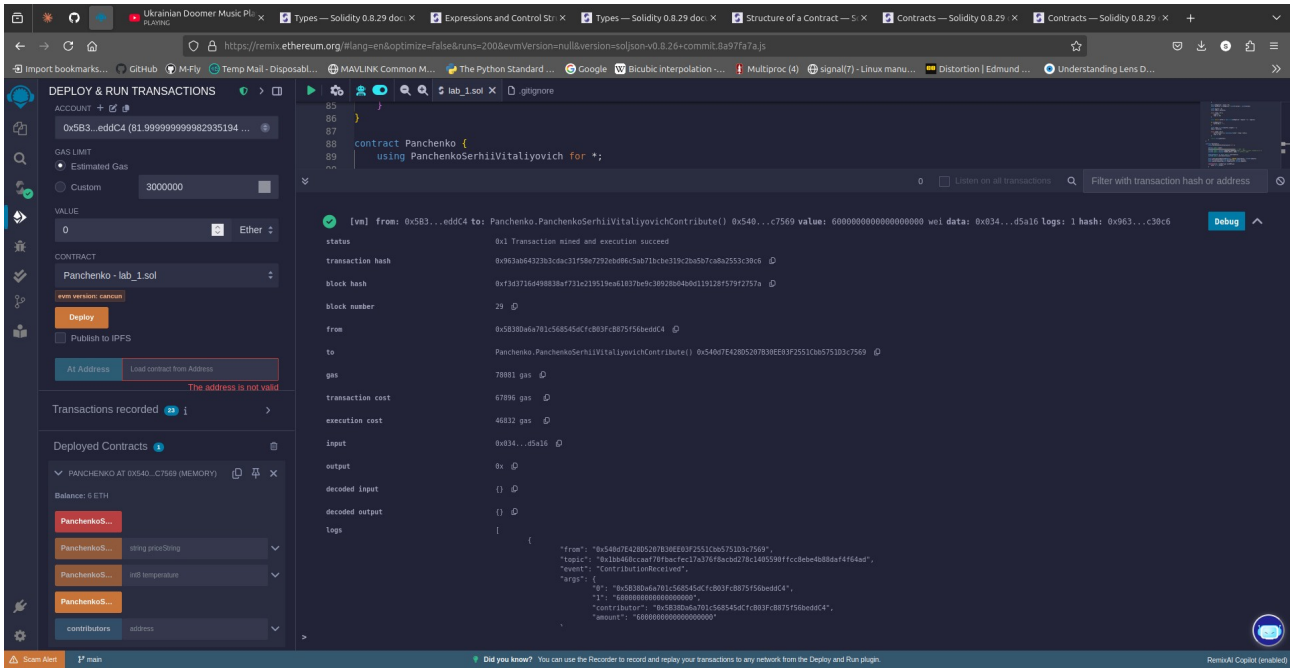


Рисунок 3.3 — контракт приймає повідомлення при правильній кількості надісланих ETH

status 0x1 Transaction mined and execution succeed

transaction hash

0x963ab64323b3cdac31f58e7292ebd06c5ab71bcbe319c2ba5b7ca8a2553c30c6

block hash

0xf3d3716d498838af731e219519ea61037be9c30928b04b0d119128f579f2757a

block number 29

from 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

to Panchenko.PanchenkoSerhiiVitaliyovichContribute()

0x540d7E428D5207B30EE03F2551Cbb5751D3c7569

gas 78081 gas

transaction cost 67896 gas

execution cost 46832 gas

input 0x034...d5a16

output 0x

decoded input {}

decoded output {}

logs [


```
{
    "from": "0x540d7E428D5207B30EE03F2551Cbb5751D3c7569",
    "topic":
"0x1bb460ccaaf70fbacfec17a376f8acbd278c1405590ffcc8ebe4b88daf4f64ad",
    "event": "ContributionReceived",
    "args": {
        "0": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
        "1": "6000000000000000000",
        "contributor": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
        "amount": "6000000000000000000"
    }
}
]
raw logs [
 {
   "logIndex": "0x1",
   "blockNumber": "0x1d",
```

"blockHash":
"0xf3d3716d498838af731e219519ea61037be9c30928b04b0d119128f579f2757a",
"transactionHash":
"0x963ab64323b3cdac31f58e7292ebd06c5ab71bcbe319c2ba5b7ca8a2553c30c6",
"transactionIndex": "0x0",
"address": "0x540d7E428D5207B30EE03F2551Cbb5751D3c7569",
"data":
"0x00000000000000000000000000000000000000000000053444835ec580000",
"topics": [
"0x1bb460ccaaf70fbacfec17a376f8acbd278c1405590ffcc8ebe4b88daf4f64ad",
"0x0000000000000000000000005b38da6a701c568545dcfcb03fcb875f56beddc4"
]
}
]

value6000000000000000000 wei

3.2 PanchenkoSerhiiVitaliyovichGetPrice

Розглянемо роботу методу PanchenkoSerhiiVitaliyovichGetPrice на рисунку 3.4.

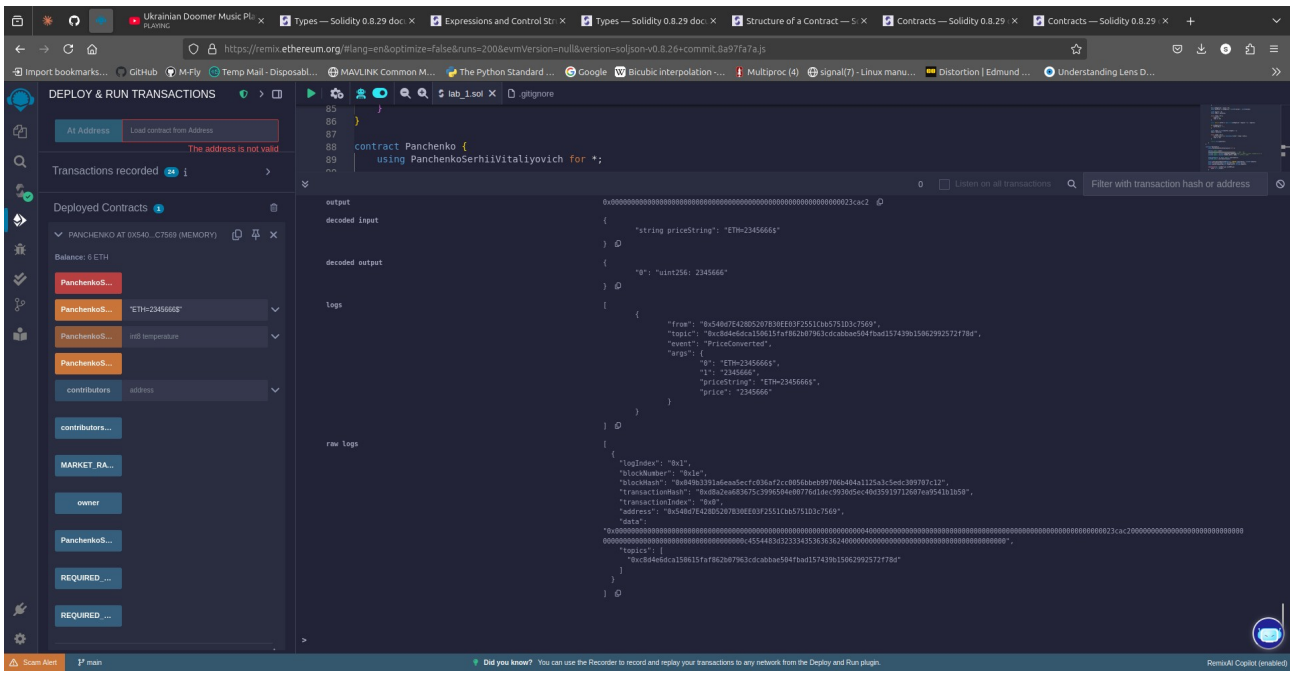


Рисунок 3.4 — робота методу PanchenkoSerhiiVitaliyovichGetPrice

[vm]

from: 0x5B3...eddC4

to: Panchenko.PanchenkoSerhiiVitaliyovichGetPrice(string) 0x540...c7569

value: 0 wei

data: 0x846...00000

logs: 1

hash: 0xd8a...b1b50

status 0x1 Transaction mined and execution succeed

transaction hash

0xd8a2ea683675c3996504e00776d1dec9930d5ec40d35919712607ea9541b1b50

block hash

0x049b3391a6eaa5ecfc036af2cc0056bbeb99706b404a1125a3c5edc309707c12

block number 30

from 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

to Panchenko.PanchenkoSerhiiVitaliyovichGetPrice(string)

0x540d7E428D5207B30EE03F2551Cbb5751D3c7569

gas 38417 gas

transaction cost 33406 gas

execution cost 11790 gas

input 0x846...00000

output

0x0000000000000000000000000000000000000000000000000000000023cac2

decoded input {

"string priceString": "ETH=2345666$"

}

decoded output {

"0": "uint256: 2345666"

}

```
logs  [
        {
                "from": "0x540d7E428D5207B30EE03F2551Cbb5751D3c7569",
                "topic":
"0xc8d4e6dca150615faf862b07963cdcabbae504fbad157439b15062992572f78d",
                "event": "PriceConverted",
                "args": {
                        "0": "ETH=2345666$",
                        "1": "2345666",
                        "priceString": "ETH=2345666$",
                        "price": "2345666"
                }
        }
]
raw logs    [
 {
   "logIndex": "0x1",
   "blockNumber": "0x1e",
   "blockHash":
"0x049b3391a6eaa5ecfc036af2cc0056bbeb99706b404a1125a3c5edc309707c12",
   "transactionHash":
"0xd8a2ea683675c3996504e00776d1dec9930d5ec40d35919712607ea9541b1b50",
   "transactionIndex": "0x0",
   "address": "0x540d7E428D5207B30EE03F2551Cbb5751D3c7569",
   "data":
"0x0000000000000000000000000000000000000000000000000000000000000040000
0000000000000000000000000000000000000000000000000000000000023cac2000000000
0000000000000000000000000000000000000000000000000000c4554483d323334
353636362400000000000000000000000000000000000000",
   "topics": [
```

```
    "0xc8d4e6dca150615faf862b07963cdcabbae504fbad157439b15062992572f78d"
  ]
 }
]
```

3.3 PanchenkoSerhiiVitaliyovichGetWeather

Розглянемо роботу методу PanchenkoSerhiiVitaliyovichGetWeather на рисунку 3.5.

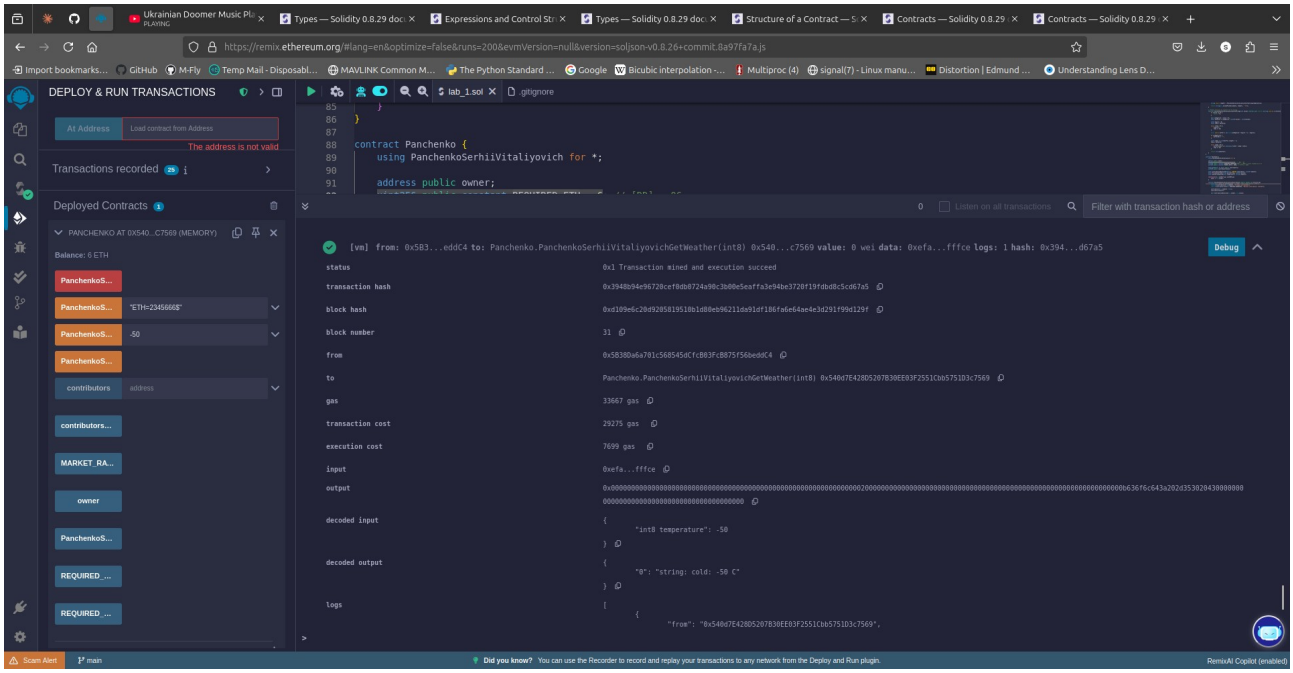


Рисунок 3.5 — робота методу PanchenkoSerhiiVitaliyovichGetWeather

```
[vm]
from: 0x5B3...eddC4
to: Panchenko.PanchenkoSerhiiVitaliyovichGetWeather(int8) 0x540...c7569
value: 0 wei
data: 0xefa...fffce
logs: 1
hash: 0x394...d67a5
status      0x1 Transaction mined and execution succeed
transaction hash
      0x3948b94e96720cef0db0724a90c3b00e5eaffa3e94be3720f19fdbd8c5cd67a5
block hash
```

0xd109e6c20d9205819510b1d80eb96211da91df186fa6e64ae4e3d291f99d129f

block number 31

from 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

to Panchenko.PanchenkoSerhiiVitaliyovichGetWeather(int8)

0x540d7E428D5207B30EE03F2551Cbb5751D3c7569

gas 33667 gas

transaction cost 29275 gas

execution cost 7699 gas

input 0xefa...fffce

output

0x0000000000000000000000000000000000000000000000000000000020000

0000000000000000000000000000000000000000000000000000000b636f6c643

a202d353020430000000000000000000000000000000000000000

decoded input {

"int8 temperature": -50

}

decoded output {

"0": "string: cold: -50 C"

}

logs [

{

"from": "0x540d7E428D5207B30EE03F2551Cbb5751D3c7569",

"topic":

"0x8a20ecedd7768ba189ffb0613fb5246b149242b0774bb8f556e06a672c6ff01a",

"event": "WeatherCommented",

"args": {

"0": -50,

"1": "cold: -50 C",

"temperature": -50,

```
                "comment": "cold: -50 C"
            }
        }
]
raw logs    [
  {
    "logIndex": "0x1",
    "blockNumber": "0x1f",
    "blockHash":
"0xd109e6c20d9205819510b1d80eb96211da91df186fa6e64ae4e3d291f99d129f",
    "transactionHash":
"0x3948b94e96720cef0db0724a90c3b00e5eaffa3e94be3720f19fdbd8c5cd67a5",
    "transactionIndex": "0x0",
    "address": "0x540d7E428D5207B30EE03F2551Cbb5751D3c7569",
    "data":
"0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffce0000000000000000000000000000000000000000000000000000000000000040000000000000000000000000000000000000000000000000000000000000000b636f6c643a202d353020430000000000000000000000000000000000000000000",
    "topics": [
      "0x8a20ecedd7768ba189ffb0613fb5246b149242b0774bb8f556e06a672c6ff01a"
    ]
  }
]
```


3.4 Публікація та валідація контракту у Sepolio

Роботу контракту можна перевірити за посиланням https://sepolia.etherscan.io/address/0x82458630b64bba37CC6f66e18568300886768ec3.

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду

(Найменування програми (документа))

Жорсткий диск

(Вид носія даних)

(Обсяг програми (документа), арк.)

Студента групи ІП-11 4 курсу

Панченка С. В

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

// Library named after the full name (PIP)
library PanchenkoSerhiiVitaliyovich {
    // Function to parse ETH price from a string
    function PanchenkoSerhiiVitaliyovichParsePrice(string memory priceString) internal pure returns (uint256) {
        bytes memory priceBytes = bytes(priceString);

        // Find the position of '='
        uint256 equalPosition = 0;
        for (uint256 i = 0; i < priceBytes.length; i++) {
            if (priceBytes[i] == '=') {
                equalPosition = i + 1;
                break;
            }
        }

        // Parse the number
        uint256 price = 0;
        for (uint256 i = equalPosition; i < priceBytes.length; i++) {
            if (priceBytes[i] >= '0' && priceBytes[i] <= '9') {
                price = price * 10 + uint256(uint8(priceBytes[i]) - 48);
            }
            if (priceBytes[i] == '$') {
                break;
            }
        }

        return price;
    }

    // Function to generate weather comment
    function PanchenkoSerhiiVitaliyovichWeatherComment(int8
```

```
temperature) internal pure returns (string memory) {
        string memory comment;

        if (temperature < 0) {
            comment = "cold: ";
        } else if (temperature < 15) {
            comment = "cool: ";
        } else if (temperature < 25) {
            comment = "warm: ";
        } else {
            comment = "hot: ";
        }

        // Convert temperature to string
        string memory tempStr =
PanchenkoSerhiiVitaliyovichIntToString(temperature);

        return string(abi.encodePacked(comment, tempStr, " C"));
    }

    // Helper function to convert int to string
    function PanchenkoSerhiiVitaliyovichIntToString(int8 value)
internal pure returns (string memory) {
        if (value == 0) {
            return "0";
        }

        bool isNegative = value < 0;
        uint8 absValue = isNegative ? uint8(-value) :
uint8(value);

        uint8 digits = 0;
        uint8 temp = absValue;

        while (temp > 0) {
            digits++;
            temp /= 10;
```

```
        }

        bytes memory buffer = new bytes(isNegative ? digits + 1 :
digits);

        if (isNegative) {
            buffer[0] = '-';
        }

        uint8 index = uint8(buffer.length) - 1;
        temp = absValue;

        while (temp > 0) {
            buffer[index--] = bytes1(uint8(48 + (temp % 10)));
            temp /= 10;
        }

        return string(buffer);
    }
}

contract Panchenko {
    using PanchenkoSerhiiVitaliyovich for *;

    address public owner;
    uint256 public constant REQUIRED_ETH = 6; // [DD] = 06
    uint256 public constant REQUIRED_ADDRESSES = 3; // [MM]/2 =
03/2 rounded up to 3
    uint256 public constant MARKET_RATE = 2004; // [YYYY] = 2004

    mapping(address => bool) public contributors;
    uint256 public contributorsCount;

    event ContributionReceived(address indexed contributor,
uint256 amount);
    event PriceConverted(string priceString, uint256 price);
    event WeatherCommented(int8 temperature, string comment);
```

```
    constructor() {
        owner = msg.sender;
    }

    function PanchenkoSerhiiVitaliyovichContribute() public
payable {
        require(msg.value == REQUIRED_ETH * 1 ether, "Must send
exactly 6 ETH");
        require(!contributors[msg.sender], "Already contributed");
        require(contributorsCount < REQUIRED_ADDRESSES, "Maximum
contributors reached");

        contributors[msg.sender] = true;
        contributorsCount++;

        emit ContributionReceived(msg.sender, msg.value);
    }

    function PanchenkoSerhiiVitaliyovichGetPrice(string memory
priceString) public returns (uint256) {
        uint256 price =
PanchenkoSerhiiVitaliyovich.PanchenkoSerhiiVitaliyovichParsePrice(
priceString);
        emit PriceConverted(priceString, price);
        return price;
    }

    function PanchenkoSerhiiVitaliyovichGetWeather(int8
temperature) public returns (string memory) {
        string memory comment =
PanchenkoSerhiiVitaliyovich.PanchenkoSerhiiVitaliyovichWeatherComm
ent(temperature);
        emit WeatherCommented(temperature, comment);
        return comment;
    }
```

```
    function PanchenkoSerhiiVitaliyovichWithdraw() public {
        require(msg.sender == owner, "Only owner can withdraw");
        require(contributorsCount == REQUIRED_ADDRESSES, "Need all
contributions first");

        payable(owner).transfer(address(this).balance);
    }

    function PanchenkoSerhiiVitaliyovichGetBalance() public view
returns (uint256) {
        return address(this).balance;
    }
}
```