

ASSIGNMENT - 3

CREATE A SMART CONTRACT FOR SALARY SLIP

WHERE BASIC =25000\$

TA = 10% of basic

DA =2000\$

HRA =30% of basic

PF = 10% of basic but must not be less then 3000\$

Find Gross salary?

Solution=>

Here are some simple calculations.

Our Basic is 25000

DA = 2000

TA is 10% of basic = 2500

HRA is 30% of basic = 7500

PF is 10% of basic(must not be less then 3000) = 3000

Overall Total of above numbers is 40000.

Now, Here a simple Solidity smart contract that calculates gross salary based on provided rules.

```
// SPDX-License-Identifier: MIT
```

```
pragma solidity ^0.8.0;
```

```
contract SalarySlip {
```

```
    uint public basic = 25000;
```

```
    uint public DA = 2000;
```

```
    // Calculate TA (10% of basic)
```

```
    function getTA() public view returns (uint) {
```

```
        return basic * 10 / 100;
```

```
    }
```

```
    // Calculate HRA (30% of basic)
```

```
    function getHRA() public view returns (uint) {
```

```
        return basic * 30 / 100;
```

```
    }
```

```
    // Calculate PF (10% of basic but at least 3000)
```

```
    function getPF() public view returns (uint) {
```

```
        uint pf = basic * 10 / 100;
```

```
        if (pf < 3000) {
```

```
            return 3000;
```

```
        }
```

```
        return pf;
```

```
    }
```

```
// Calculate Gross Salary (sum of all earnings)
function getGrossSalary() public view returns (uint) {
    return basic + getTA() + DA + getHRA();
}

// Calculate Net Salary (Gross Salary - PF)
function getNetSalary() public view returns (uint) {
    return getGrossSalary() - getPF();
}
}
```

How It Works

basic, DA are set as per values.

getTA(), getHRA(), and getPF() fetch the correct allowance amounts.

getPF() implements the minimum limit you specified.

getGrossSalary() adds up all components for the total.

Summary of Untitled3 Document

The document, titled “ASSIGNMENT - 3,” focuses on a single task: **creating a Solidity smart contract to calculate a salary slip based on a specific set of financial rules.**

1. The Core Problem and Financial Rules

The document presents a hypothetical scenario for a salary slip calculation, defining the following components and rules:

Component	Value/Rule
Basic Salary (Basic)	\$25,000
Travel Allowance (TA)	10% of Basic
Dearness Allowance (DA)	\$2,000
House Rent Allowance (HRA)	30% of Basic
Provident Fund (PF)	10% of Basic, but must not be less than \$3,000
Goal	Find the Gross Salary and implement the calculation in a smart contract.

2. Manual Calculation and Verification

The document first performs a manual calculation based on the rules:

- **Basic:** \$25,000
- **DA:** \$2,000
- **TA (10% of \$25,000):** \$2,500
- **HRA (30% of \$25,000):** \$7,500

- **PF (10% of 25,000 is 2,500, but minimum is \$3,000): \$3,000**

3. The Smart Contract Implementation

The main argument is that a **Solidity smart contract** can be used to automate and enforce these financial rules on a blockchain.

The document provides a complete `SalarySlip` contract with the following functions:

- `getTA()` : Calculates TA (10% of basic).
- `getHRA()` : Calculates HRA (30% of basic).
- `getPF()` : Calculates PF, correctly implementing the conditional logic to ensure the returned value is **at least 3000** (the minimum threshold).
- `getGrossSalary()` : Calculates the total earnings (Basic + TA + DA + HRA).
- `getNetSalary()` : Calculates the final take-home pay (Gross Salary - PF).

In summary, the document serves as a solution to a technical assignment, demonstrating how to translate a set of specific, conditional financial rules into a functional and verifiable **Solidity smart contract** for calculating a salary slip.