BCIT FIRST YEAR SECTION D

MADE BY: ANZAL(CT-25177), ADAM(CT-25190) AND AREEBA(CT25158)

Payroll Management System

Index

1. **Title Page**
2. **Abstract**
3. **Introduction**
4. **Project Title**
5. **Project Description**
6. **Project Methodology**

   6.1 Dataset/Input Data

   6.2 Tools and Technologies

   6.3 Algorithm

   6.4 Objectives

   6.5 Flowchart

   6.6 Expected Outcomes

   6.7 Goals

7. **Justification**
8. **Industrial Product Potential**
9. **Program Implementation**
10. **Memory & Arrays**

11. **Input Collection**
    11.1 Employee Details
    11.2 Loop Implementation
    11.3 Input Example Tables
12. **Detailed Calculations Explanation**
    12.1 Basic Pay
    12.2 Overtime Pay
    12.3 Allowances
    12.4 Performance Bonus

    12.5 Tenure Bonus
    12.6 Unpaid Deduction

    12.7 Gross Pay
    12.8 Tax Deduction

12.9 Net Pay
12.10 Batch Processing & Efficiency
12.11 Sample Payroll Table

13. **Discussion**
    13.1 Conditional Logic Explanation
    13.2 Array Efficiency & Batch Processing
    13.3 Error Reduction & Transparency
    13.4 Edge Cases & Considerations
14. **Advantages of the System**
15. **Limitations**
16. **Future Enhancements**
17. **Real-World Relevance**
18. **Conclusion**

2.Abstract

The **Payroll Management System** is a C-based software application designed to automate employee salary calculations for organizations of any size. This system integrates multiple payroll components, including **basic pay, overtime, allowances, performance bonuses, tenure-based incentives, unpaid leave deductions, and tax calculations**, into a single automated framework.

Manual payroll processing is prone to errors, time-consuming, and difficult to scale. By leveraging **structured programming concepts, arrays for batch processing, and conditional logic**, the system ensures accurate computation and transparent salary reporting for multiple employees simultaneously.

The system also demonstrates **modular programming practices**, separating input collection, payroll computation, and output generation into distinct phases. It provides **clear data mapping**, linking all inputs and outputs for each employee via array indices, thereby maintaining **data integrity** and reducing potential errors.

3.Introduction

Payroll management is a critical aspect of any organization. Manual calculations are:

- Prone to errors
- Time-consuming
- Difficult to scale efficiently

The Payroll Management System addresses these challenges by **automating salary calculation and reporting**, integrating multiple variables, and ensuring **accuracy for multi-employee payroll processing**.

4.Project Title:

Payroll Management System

5. Project Description:

The Payroll Management System is a computerized software-based solution that calculates employees' monthly salaries by processing their salary structure, allowances, deductions, working hours, tenure bonuses, unpaid leaves and tax amounts. Payroll automation eliminates manual calculation errors, ensures accuracy, generates salary slips and efficiently maintains employee payment records.

6. Project Methodology:

6.1 Dataset / Input Data:

* Employee ID, Name

* Basic Salary

* Allowances (House Rent, Medical, Transport)

* Deductions (Tax, Fund)
* Hours Worked
* Unpaid Time Off
* Bonus Rate per Hour
* Tenure (Years of Service)

6.2 Tools and Technologies:

* Language: C Programming
* IDE: Dev C++ / CodeBlocks / Visual Studio Code
* Platform: Windows OS
* Compiler: GCC

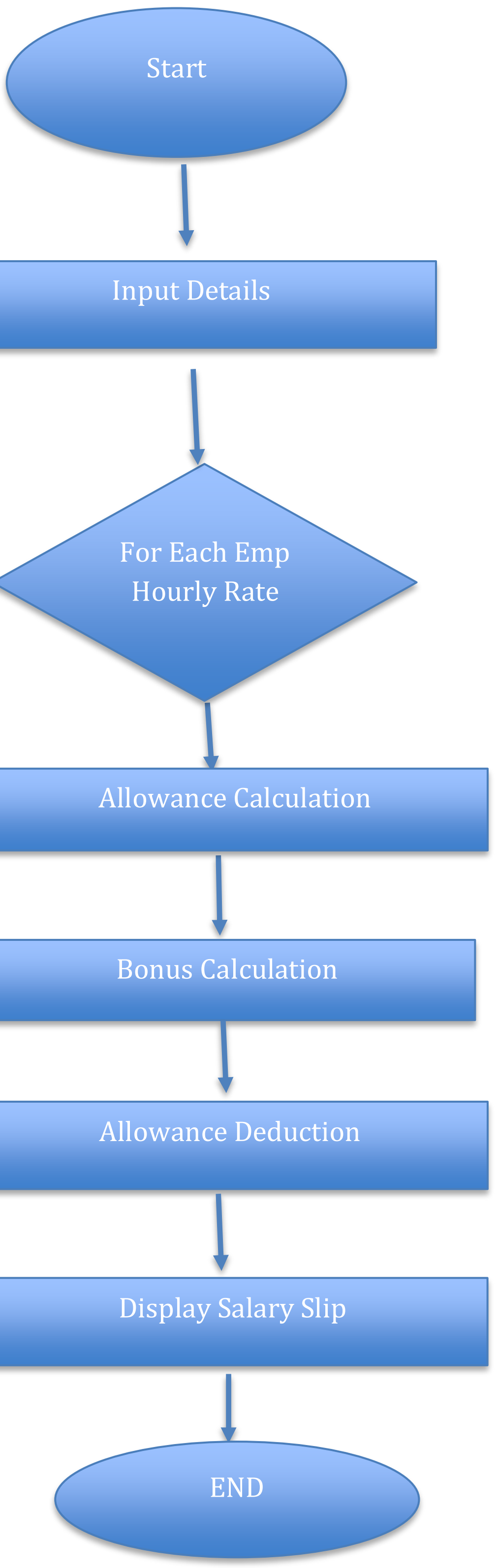
6.3 Algorithm:

1. Start
2. Input employee details
3. Calculate total allowances
4. Calculate performance bonus based on hours worked
5. Calculate tenure bonus depending on years of service
6. Calculate unpaid leave deduction based on hourly rate
7. Calculate gross salary
8. Subtract deductions from gross salary
9. Display payroll slip with final net pay
10. End

6.4 Objectives:

* Automate salary calculation process
* Provide accurate and fast payroll slip generation
* Reduce human calculation error
* Include bonus and deduction logic dynamically

6.5 Flowchart:



6.6 Expected Outcomes:

* Accurate payroll slip generation for each employee

* Automatic net salary computation
* Professional structured salary breakdown

6.7 Goals:

* Build a clear logical payroll model
* Make calculation transparent & replicable
* Implement in modular C program format

7. Justification – Why It Is a Complex Computing Problem:

Payroll contains multiple salary influencing variables. Each factor can change calculation logic. Accounting rules, working hours, deduction criteria & tenure bonuses require conditional logic, mathematical processing and systematic breakdown which cannot be reliably handled through manual repetitive calculations.

8. Industrial / Commercial Product Potential:

This Payroll System can be used in small offices, colleges, software houses and business organizations to prepare monthly salary sheets. It can be expanded into a full HR management module by adding attendance system, employee database storage, multi employee batch payroll processing and salary exporting to PDF / CSV files.

9. Program Implementation:

The following C program calculates payroll for multiple employees, including basic pay, overtime pay, gross pay, tax deduction, and net pay.

```
#include <stdio.h>

int main() {
   int n, i;
   int empID[50], tenure[50];
   char name[50][50];
   float hourlyRate[50], hoursWorked[50], overtimeHours[50],
unpaidHours[50];
   float houseAllowance[50], medicalAllowance[50],
transportAllowance[50];
```

```
   float basicPay[50], overtimePay[50], allowanceTotal[50],
performanceBonus[50];
   float tenureBonus[50], unpaidDeduction[50], grossPay[50], tax[50],
netPay[50];
   printf("===== PAYROLL SYSTEM =====\n");
   printf("Enter number of employees: ");
   scanf("%d", &n);
   for (i = 0; i < n; i++) {
      printf("\n--- Enter details for Employee %d ---\n", i + 1);
      printf("Enter Employee ID: ");
      scanf("%d", &empID[i]);
      printf("Enter Employee Name: ");
      scanf(" %[^\n]", name[i]);
      printf("Enter Hourly Rate: ");
      scanf("%f", &hourlyRate[i]);
      printf("Enter Hours Worked: ");
      scanf("%f", &hoursWorked[i]);
      printf("Enter Overtime Hours: ");
      scanf("%f", &overtimeHours[i]);
      printf("Enter Unpaid Leave Hours: ");
      scanf("%f", &unpaidHours[i]);
      printf("Enter House Allowance: ");
      scanf("%f", &houseAllowance[i]);
      printf("Enter Medical Allowance: ");
      scanf("%f", &medicalAllowance[i]);
      printf("Enter Transport Allowance: ");
      scanf("%f", &transportAllowance[i]);
      printf("Enter Tenure (years of service): ");
      scanf("%d", &tenure[i]);}
      // Calculations
      basicPay[i] = hoursWorked[i] * hourlyRate[i];
      overtimePay[i] = overtimeHours[i] * (hourlyRate[i] * 1.5);
      allowanceTotal[i] = houseAllowance[i] + medicalAllowance[i] +
transportAllowance[i];
      performanceBonus[i] = (hoursWorked[i] > 160) ? (hoursWorked[i]
- 160) * (hourlyRate[i] * 0.25) : 0;
      tenureBonus[i] = tenure[i] * (hourlyRate[i] * 10);
      unpaidDeduction[i] = unpaidHours[i] * hourlyRate[i];
```

```
        grossPay[i] = basicPay[i] + overtimePay[i] + allowanceTotal[i] +
performanceBonus[i] + tenureBonus[i];
        grossPay[i] -= unpaidDeduction[i];
        tax[i] = grossPay[i] * 0.10;
        netPay[i] = grossPay[i] - tax[i];
    }
    printf("\n===== PAYROLL SUMMARY =====\n");
    for (i = 0; i < n; i++) {
        printf("\nEmployee %d\n", i + 1);
        printf("ID: %d\n", empID[i]);
        printf("Name: %s\n", name[i]);
        printf("Basic Pay: %.2f\n", basicPay[i]);
        printf("Overtime Pay: %.2f\n", overtimePay[i]);
        printf("Total Allowances: %.2f\n", allowanceTotal[i]);
        printf("Performance Bonus: %.2f\n", performanceBonus[i]);
        printf("Tenure Bonus: %.2f\n", tenureBonus[i]);
        printf("Unpaid Leave Deduction: %.2f\n", unpaidDeduction[i]);
        printf("Gross Pay: %.2f\n", grossPay[i]);
        printf("Tax (10%%): %.2f\n", tax[i]);
        printf("Net Pay: %.2f\n", netPay[i]);
        printf("--------------------------\n");
    }
    printf("===== END OF PAYROLL =====\n");
    return 0;
}
```

10.Memory & Arrays

Arrays are used to store multiple employees' data simultaneously. Each index represents a unique employee.

Example:

```
float netPay[50]; // Stores net pay for 50
employees
```

Advantages:

- **Data Integrity:** Each employee's data is isolated by index
- **Batch Processing:** All calculations performed using loops
- **Efficiency:** Single set of instructions for multiple employees

[Insert memory mapping diagram showing arrays storing employee data]

11.Input Collection

11.1 Details Collected:

- Employee ID, Name
- Hours Worked, Hourly Rate
- Overtime, Unpaid Hours
- Allowances (House, Medical, Transport)
- Tenure (Years of Service)

11.2 Loop Implementation:

```
for (i = 0; i < n; i++) {
    scanf("%d", &empID[i]);
    scanf(" %[^\n]", name[i]);
    scanf("%f", &hourlyRate[i])...}
```

This ensures **structured input for all employees**. Arrays link all attributes of a single employee together.

11.3 Example Table: Input Sample

Employee	Hours Worked	Hourly Rate	Overtime	Tenure
1	160	200	10	3
2	150	180	5	5

12. Detailed Calculations Explanation

12.1. Basic Pay

Formula: Basic Pay = Hours Worked × Hourly Rate
Example: 160 × 200 = 32,000

12.2. Overtime Pay

Formula: Overtime Hours × 1.5 × Hourly Rate
Example: 10 × 200 × 1.5 = 3,000

12.3. Allowances

Formula: House + Medical + Transport
Example: 5,000 + 2,000 + 1,500 = 8,500

12.4. Performance Bonus

Conditional Formula: (Hours Worked > 160) ? (Hours-160) × (Rate × 0.25) : 0
Example: 10 × 200 × 0.25 = 500

12.5. Tenure Bonus

Formula: Years × Rate × 10
Example: 3 × 200 × 10 = 6,000

12.6. Unpaid Deduction

Formula: Unpaid Hours × Rate
Example: 5 × 200 = 1,000

12.7. Gross Pay

Sum of all earnings minus unpaid deduction
Example: 49,000

12.8. Tax Deduction

Formula: 10% of Gross Pay
Example: 49,000 × 0.10 = 4,900

12.9. Net Pay

Gross Pay – Tax = Final take-home salary
Example: 44,100

12.10. Batch Processing

Arrays and loop iteration allow **all employees' payrolls to be calculated simultaneously**, reducing errors and increasing efficiency.

12.11. Example Table: Full Payroll

Component	**Employee 1**	**Employee 2**
Ba1sic Pay	32000	27000
Overtime	3000	1350
Allowances	8500	7500
Bonus	500	0
Tenure	6000	9000
Unpaid Deduction	1000	1800
Gross Pay	49000	43050
Tax	4900	4305
Net Pay	44100	38745

13. Discussion

13.1 Conditional Logic Explanation

The Payroll Management System heavily relies on **conditional logic** to calculate bonuses and deductions. For example, the **Performance Bonus** is only applied if an employee works more than 160 hours in a month. This is implemented using a **ternary operator** in C:

```
performanceBonus[i] = (hoursWorked[i] > 160) ?
                      (hoursWorked[i] - 160) *
(hourlyRate[i] * 0.25) : 0;
```

Explanation:

- The system checks if hoursWorked[i] > 160.
- If **true**, it calculates 25% of the excess hours.
- If **false**, the bonus is set to zero.

This ensures **fair performance compensation** while maintaining efficiency, as the condition is evaluated for each employee in a batch.

13.2 Array Efficiency & Batch Processing

Arrays allow the system to store multiple employees' data simultaneously. Each employee's data is stored at the same index across all arrays (e.g., hoursWorked[i], hourlyRate[i], netPay[i]).

Advantages:

- **Batch Processing:** Payroll for multiple employees is calculated in a single loop.
- **Data Integrity:** Each employee's inputs and outputs remain linked via the array index.
- **Reduced Repetition:** The same calculation logic applies to all employees, improving maintainability.

Example:

```
for(i = 0; i < n; i++) {
    basicPay[i] = hoursWorked[i] * hourlyRate[i];
    overtimePay[i] = overtimeHours[i] * (hourlyRate[i] * 1.5);
    ...
}
```

This loop ensures that all payroll components are processed **simultaneously and efficiently**.

13.3 Error Reduction & Transparency

Manual payroll calculation is prone to errors, especially with:

- Complex allowances
- Tenure bonuses
- Unpaid leave deductions

By automating these calculations, the system ensures:

- Accurate computation of all pay components
- Transparency in how gross pay, deductions, and net pay are calculated
- Easier auditing, as all data is stored in arrays and displayed systematically

13.4 Edge Cases & Considerations

The system is designed to handle various **edge cases**, including:

- Unpaid leave exceeding total hours
- Overtime hours set to zero
- Employees with no allowances or bonuses

These are managed using **conditional statements**, ensuring that the system **does not crash** and generates valid outputs for all employees.

14. Advantages of the System

- Reduces human errors
- Saves time
- Supports batch processing
- Transparent salary calculation

- Scalable for HR systems

[Insert Payroll comparison chart: Manual vs Automated]

15. Limitations

- Maximum 50 employees (fixed arrays)
- Console-based interface
- Hard-coded tax and bonus rules
- No database integration

[Insert Limitation diagram showing fixed array size]

16. Future Enhancements

- Dynamic arrays or linked lists for unlimited employees
- GUI interface for better usability
- Configurable tax and bonus rules
- Attendance integration
- Export payroll to PDF/Excel

17. Real-World Relevance

- Automates payroll for businesses of all sizes
- Reduces operational costs
- Forms the foundation for professional HR systems

18. Conclusion:

The Payroll Management System streamlines employee salary management by integrating multiple salary-related parameters into a single automated platform. It reduces human errors, speeds up payroll processing, and ensures accurate computation of allowances, bonuses, deductions, and net pay. This system is practical for organizations of any size and provides a foundation for further enhancements such as attendance tracking, reporting, and integration with financial systems. Ultimately, it enhances operational efficiency and ensures transparency and reliability in employee salary management.