Name: Arpita Dinesh Singh
2nd-year B.tech
Computer engineering
BATCH A
Roll No:231071005

DESIGN ANALYSIS OF ALGORITHM

SALARY CALCULATION OF 2000 EMPLOYEES

CODE:

CODE FOR BOTH SALARY CALCULATION AND FIND=MIN,MAX USING COHESION AND COUPLING:

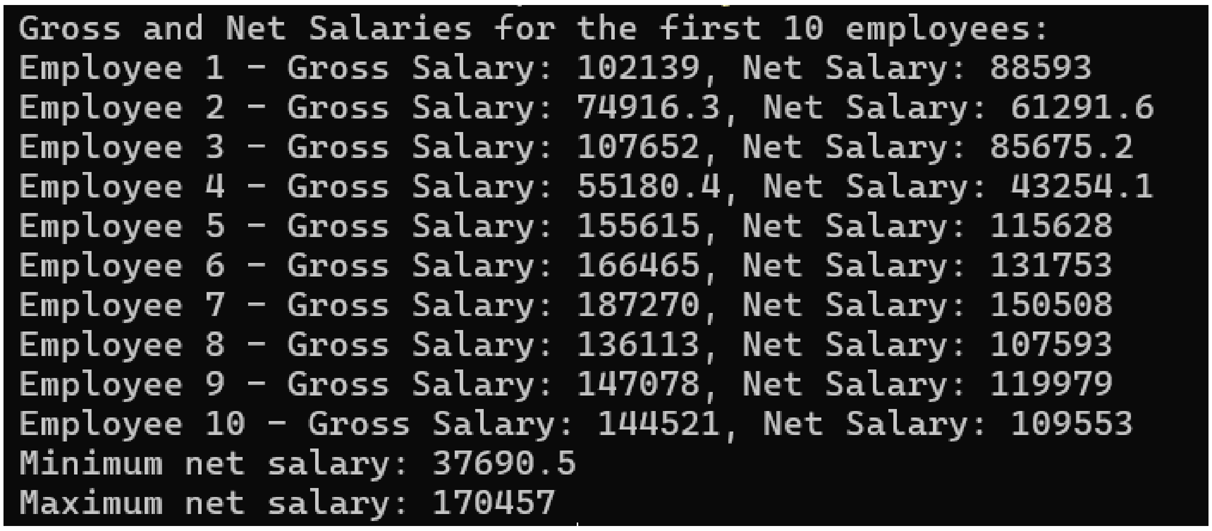
```
#include <iostream>
#include <vector>
#include <fstream>
#include <sstream>
#include <tuple>
#include <limits>
#include <iomanip>

using namespace std;

class SalaryCalculator {
private:
    double basicSalary;
    double bonus;
    static constexpr double conveyanceAllowance =
1500.0;
```

```
    static constexpr double hraPercentage = 50.0;
    static constexpr double daPercentage = 10.0;
    static constexpr double epfPercentage = 12.0;
    static constexpr double estPercentage = 0.75;

public:
    SalaryCalculator(double basic, double bon)
        : basicSalary(basic), bonus(bon) {}

    double calculateHRA() const {
        return (hraPercentage / 100) * basicSalary;
    }

    double calculateDA() const {
        return (daPercentage / 100) * basicSalary;
    }

    double calculateEPF() const {
        return (epfPercentage / 100) * basicSalary;
    }

    double calculateEST() const {
        return (estPercentage / 100) * basicSalary;
    }

    double calculateGrossSalary() const {
        return basicSalary + calculateHRA() +
conveyanceAllowance + calculateDA() + bonus;
    }
    double calculateNetSalary(double taxPercentage)
const {
        double grossSalary = calculateGrossSalary();
        double totalDeductions = calculateEPF() +
calculateEST();
        double netSalaryBeforeTax = grossSalary -
totalDeductions;
        double taxAmount = (netSalaryBeforeTax *
taxPercentage) / 100;
        return netSalaryBeforeTax - taxAmount;
    }

    static void findMinMax(const vector<double>& arr, int
left, int right, double &minSalary, double &maxSalary) {
        if (left > right) {
            throw invalid_argument("Invalid range for
min/max computation.");
        }
        if (left == right) {
            minSalary = maxSalary = arr[left];
        } else if (right == left + 1) {
            minSalary = min(arr[left], arr[right]);
            maxSalary = max(arr[left], arr[right]);
        } else {
            int mid = (left + right) / 2;
            double minLeft, maxLeft, minRight, maxRight;
            findMinMax(arr, left, mid, minLeft, maxLeft);
            findMinMax(arr, mid + 1, right, minRight,
maxRight);
            minSalary = min(minLeft, minRight);
            maxSalary = max(maxLeft, maxRight);
        }
    }
};

void readDataFromCSV(const string& filename,
vector<tuple<double, double, double>>& data) {
    ifstream file(filename);
    string line;

    if (file.is_open()) {
        getline(file, line); // Skip header row if present
        while (getline(file, line)) {
            stringstream ss(line);
            double basicSalary, bonus, taxPercentage;
            char comma;

            if (ss >> basicSalary >> comma >> bonus >>
comma >> taxPercentage) {
                data.emplace_back(basicSalary, bonus,
taxPercentage);
            } else {
                cerr << "Error parsing line: " << line << endl;
            }
        }
        file.close();
    } else {
        cerr << "Unable to open file for reading." << endl;
    }
}

void writeSalariesToCSV(const string& filename, const
vector<double>& salaries) {
    ofstream file(filename);
    if (file.is_open()) {
        for (size_t i = 0; i < salaries.size(); ++i) {
            file << fixed << setprecision(2) << salaries[i];
            if (i < salaries.size() - 1) file << ",";
        }
        file.close();
    } else {
        cerr << "Unable to open file for writing." << endl;
    }
}

int main() {
    vector<tuple<double, double, double>> data;
    vector<double> netSalaries;
    vector<double> grossSalaries;

    // Read data from CSV
    readDataFromCSV("salaries_data.csv", data);

    // Calculate salaries
    for (const auto& record : data) {
        double basicSalary = get<0>(record);
        double bonus = get<1>(record);
        double taxPercentage = get<2>(record);

        SalaryCalculator salaryCalc(basicSalary, bonus);
        double grossSalary =
salaryCalc.calculateGrossSalary();
        double netSalary =
salaryCalc.calculateNetSalary(taxPercentage);
        netSalaries.push_back(netSalary);
        grossSalaries.push_back(grossSalary);
    }

    // Find min and max net salaries
    double minNetSalary =
numeric_limits<double>::max();
    double maxNetSalary =
numeric_limits<double>::lowest();

    if (!netSalaries.empty()) {
        try {
            SalaryCalculator::findMinMax(netSalaries, 0,
netSalaries.size() - 1, minNetSalary, maxNetSalary);
        } catch (const exception& e) {
            cerr << "Exception: " << e.what() << endl;
            return 1;
        }

        // Display salaries for the first 10 employees
        cout << "Gross and Net Salaries for the first 10
employees:" << endl;
        for (size_t i = 0; i < min(static_cast<size_t>(10),
netSalaries.size()); ++i) {
            cout << "Employee " << (i + 1) << " - Gross
Salary: " << fixed << setprecision(2) << grossSalaries[i]
                 << ", Net Salary: " << fixed << setprecision(2)
<< netSalaries[i] << endl;
        }

        // Display minimum and maximum net salaries
        cout << "Minimum net salary: " << fixed <<
setprecision(2) << minNetSalary << endl;
        cout << "Maximum net salary: " << fixed <<
setprecision(2) << maxNetSalary << endl;
    } else {
        cout << "No salaries available to compute." <<
endl;
    }

    // Write net salaries to CSV
    writeSalariesToCSV("net_salaries.csv", netSalaries);

    return 0;
}
```

CODE FOR CSV FILE:

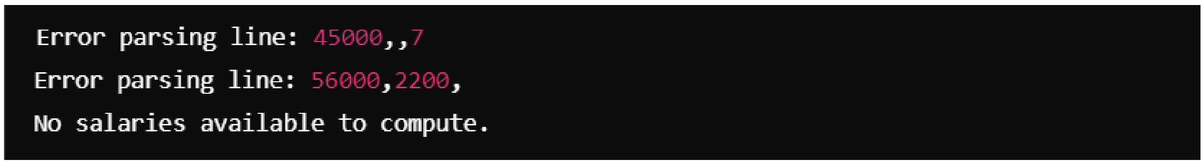
```
import csv
import random

# Parameters
num_employees = 2000
filename = 'salaries_data.csv'

# Generate random employee data
data = []
for _ in range(num_employees):
    basic_salary = round(random.uniform(30000, 120000), 2)
    bonus = round(random.uniform(1000, 5000), 2)
    tax_percentage = round(random.uniform(5, 20), 2)
    data.append([basic_salary, bonus, tax_percentage])

# Write data to CSV file
with open(filename, mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(['basicSalary', 'bonus', 'taxPercentage'])
    writer.writerows(data)

print(f"CSV file '{filename}' created with {num_employees}
employee records.")
```

SCREENSHOT:

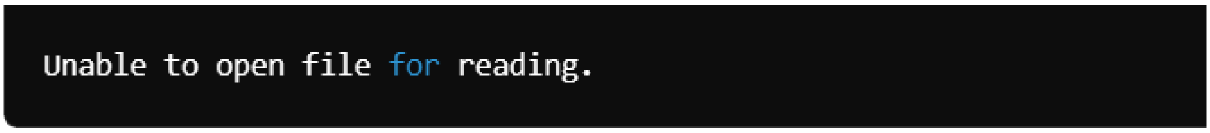


```
Gross and Net Salaries for the first 10 employees:
Employee 1 - Gross Salary: 102139, Net Salary: 88593
Employee 2 - Gross Salary: 74916.3, Net Salary: 61291.6
Employee 3 - Gross Salary: 107652, Net Salary: 85675.2
Employee 4 - Gross Salary: 55180.4, Net Salary: 43254.1
Employee 5 - Gross Salary: 155615, Net Salary: 115628
Employee 6 - Gross Salary: 166465, Net Salary: 131753
Employee 7 - Gross Salary: 187270, Net Salary: 150508
Employee 8 - Gross Salary: 136113, Net Salary: 107593
Employee 9 - Gross Salary: 147078, Net Salary: 119979
Employee 10 - Gross Salary: 144521, Net Salary: 109553
Minimum net salary: 37690.5
Maximum net salary: 170457
```

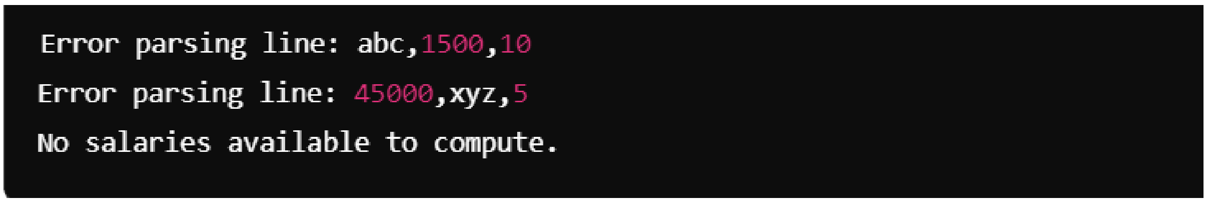
TEST CASES USE:



```
Error parsing line: 45000,,7
Error parsing line: 56000,2200,
No salaries available to compute.
```



```
Unable to open file for reading.
```



```
Error parsing line: abc,1500,10
Error parsing line: 45000,xyz,5
No salaries available to compute.
```

CONCLUSION:

This project efficiently calculates employee salaries and finds the minimum and maximum net salaries using both linear and divide and conquer methods. The linear approach ensures quick salary computation, while the divide and conquer technique provides an efficient way to identify salary minimum and maximum.