# DAA LABORATORY 4

**Name:Dhruv Panchal**

**Roll NO;231070038**

**SY Btech Comp eng.**

**Aim:** Writing an algorithm to find gross and net salary of employees.

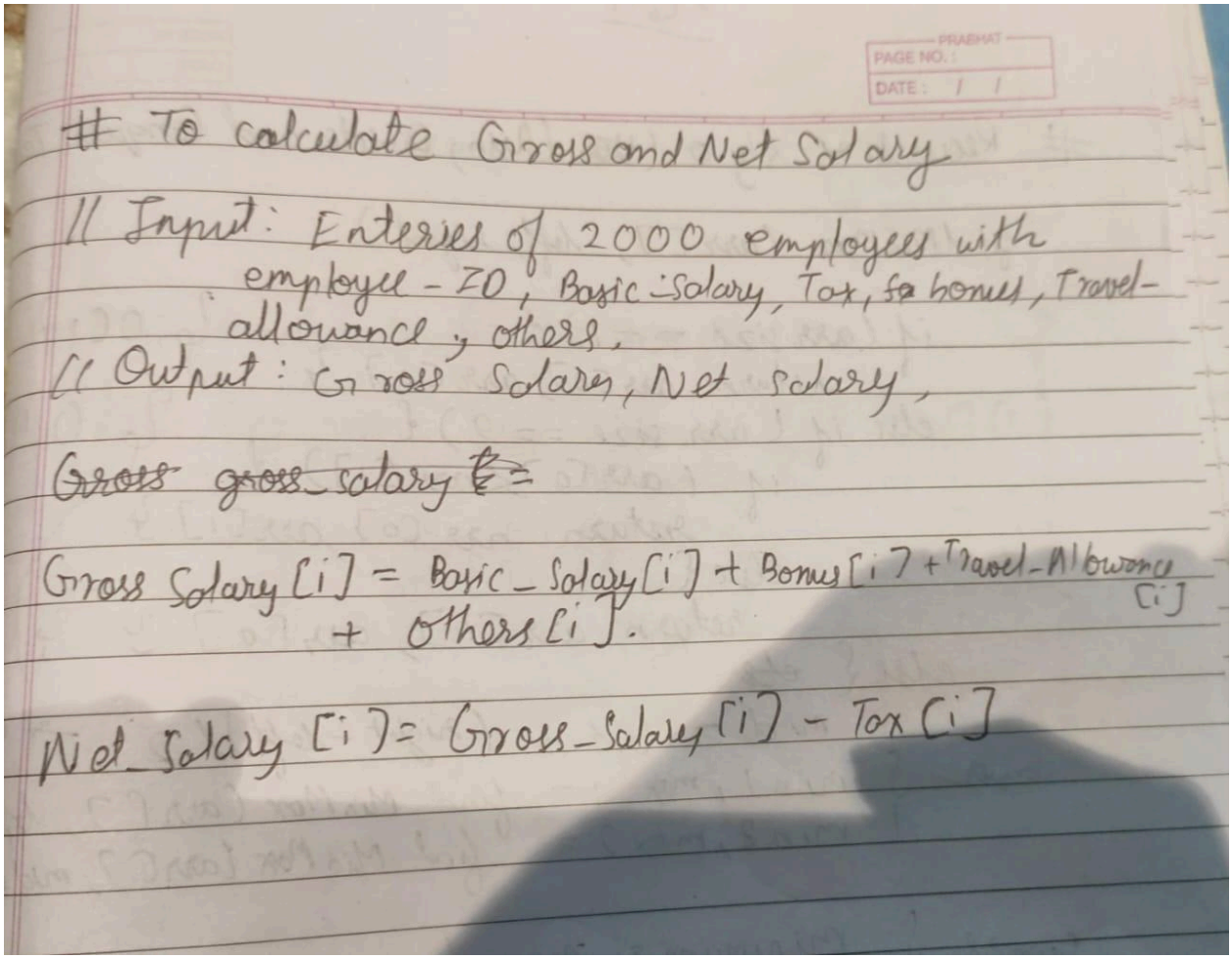
ABC co. ltd. has 2000 employees.

your task is to calculate each employees salary and find employee with minimum salary and maximum salary.

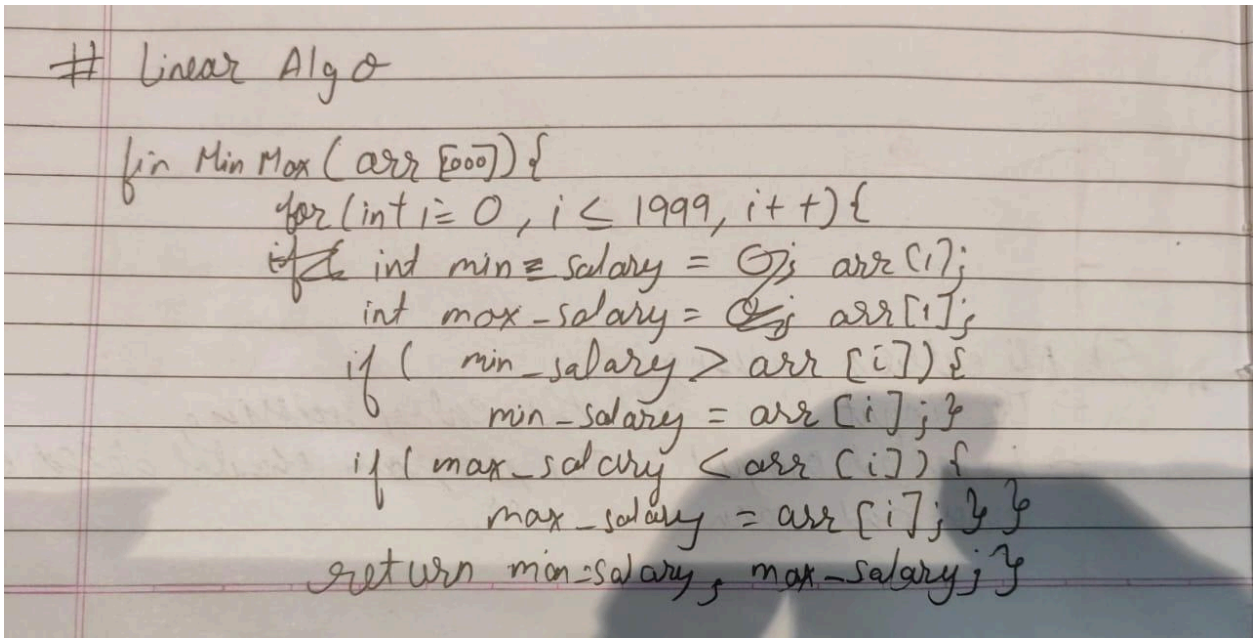
Do the above task using divide and conquer technique.

Find the improvement in the complexity using divide and conquer method.

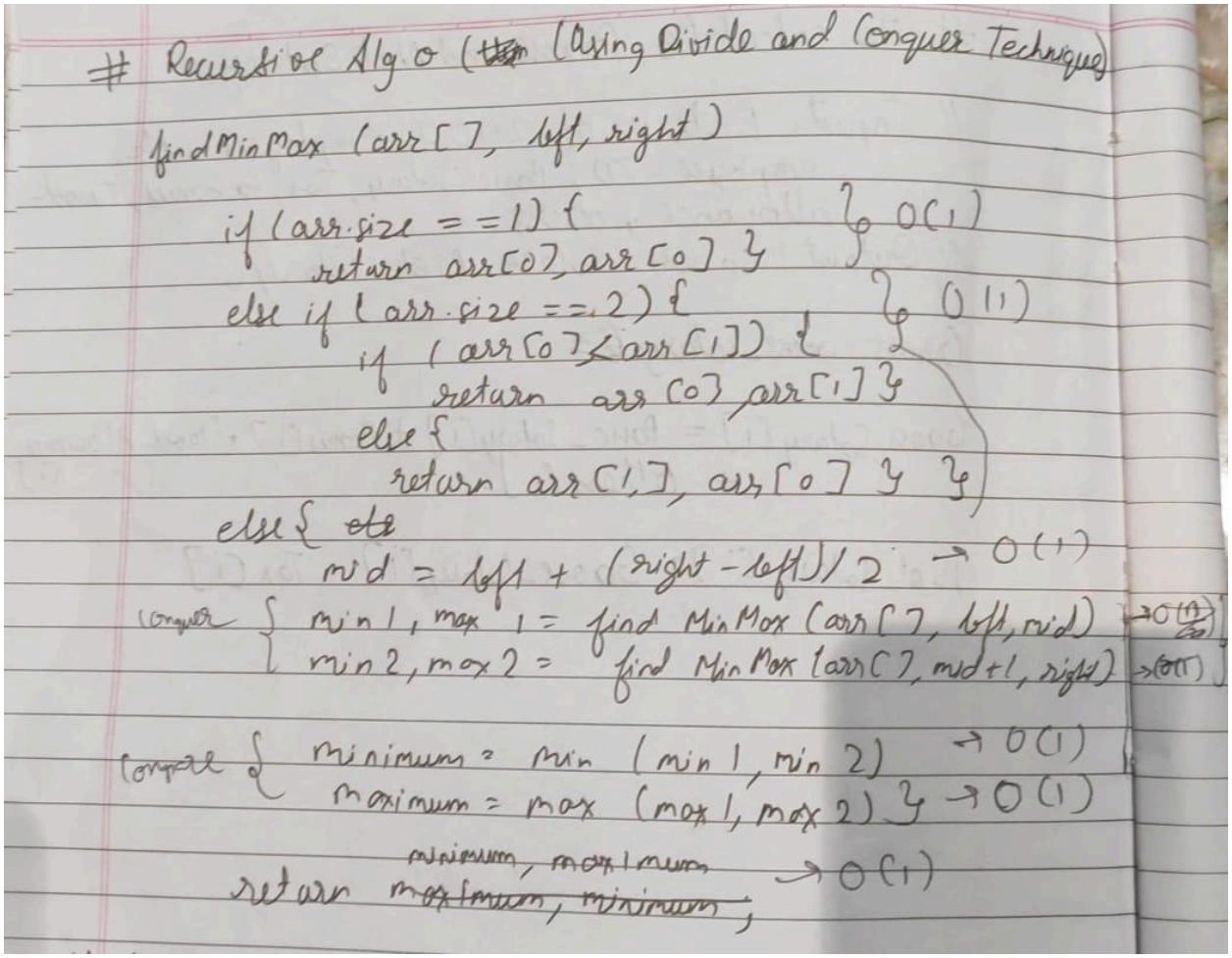
ALGORITHM TO FInd Gross and Net Salary:



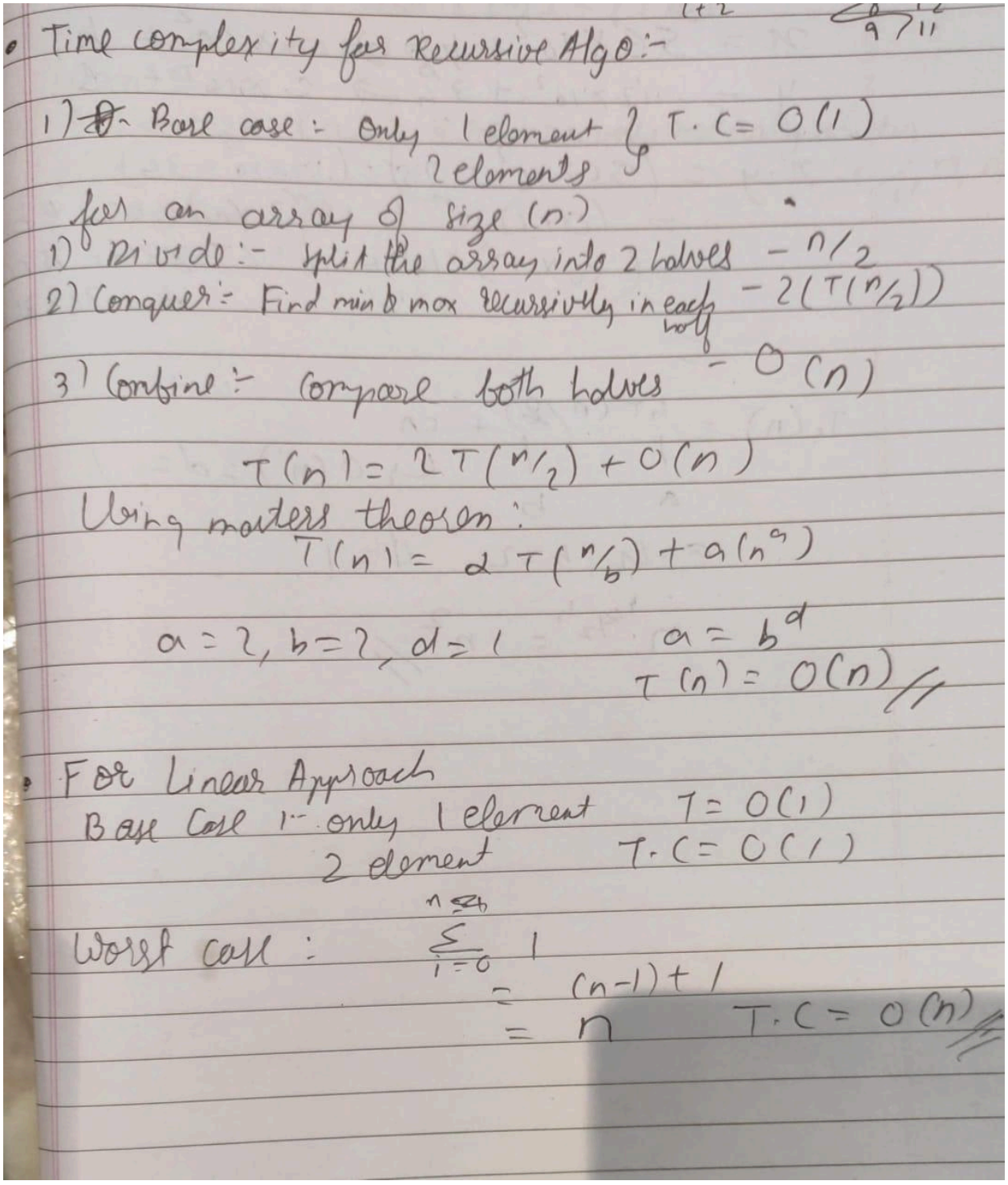

ALGORITHM TO FIND MIN MAX USING LINEAR APPROACH:



ALGORITHM TO FIND MIN MAX USING DIVIDE & CONQUER ALGO:




TIME COMPLEXITY:




CODE:

```
import csv
```

```
class Employee:

    def __init__(self, emp_id, basic_salary, tax, travel_allowance, bonus,
      others):

        self.id = self.validate_int(emp_id, 'Employee ID')

        self.basic_salary = self.validate_float(basic_salary, 'Basic
      Salary')

        self.tax = self.validate_tax(self.validate_float(tax, 'Tax'))

        self.travel_allowance = self.validate_float(travel_allowance,
      'Travel Allowance')

        self.bonus = self.validate_float(bonus, 'Bonus')

        self.others = self.validate_float(others, 'Others')

        self.gross_salary = 0

        self.net_salary = 0

        self.valid = self.compute_salaries()


    def validate_int(self, value, field_name):

        if value.strip() == '':  # Check for missing value

            print(f"Missing value for {field_name}. Defaulting to 0.")

            return 0

        try:

            return int(value)

        except ValueError:

            raise ValueError(f"Datatype mismatch detected: {field_name}
      should be an integer, but got '{value}'.")
```

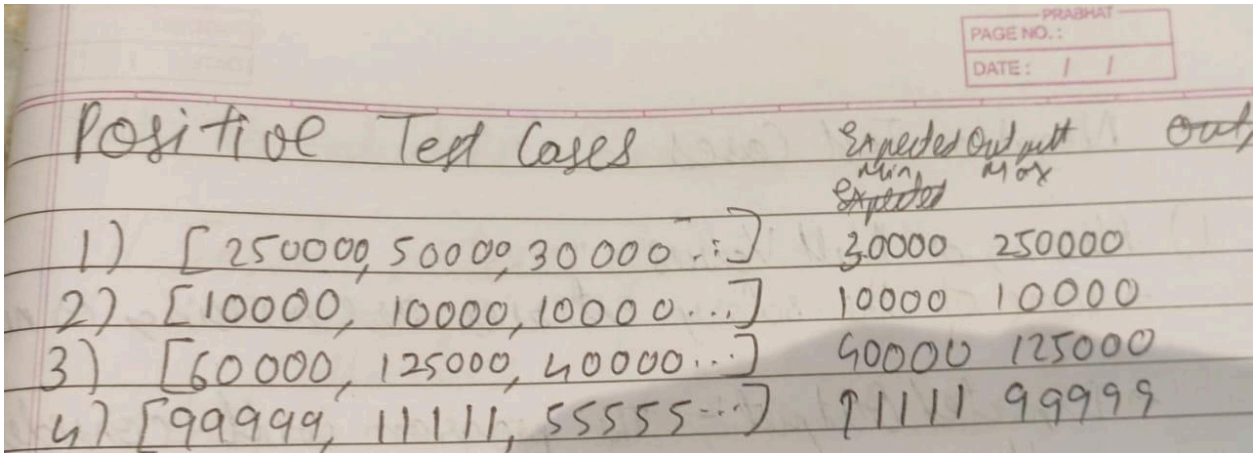
```
    def validate_float(self, value, field_name):
        if value.strip() == '':  # Check for missing value
            print(f"Missing value for {field_name}. Defaulting to 0.")
            return 0.0
        try:
            return float(value)
        except ValueError:
            raise ValueError(f"Datatype mismatch detected: {field_name}
      should be a float, but got '{value}'.")


    def validate_tax(self, tax):
        if tax < 0:
            print(f"Warning: Negative tax value detected for Employee ID
      {self.id}. Tax has been set to 0.")
            return 0
        return tax


    def compute_salaries(self):
        if self.basic_salary <= 0:
            print(f"Skipping Employee ID {self.id} due to invalid Basic
      Salary ({self.basic_salary}).")
            return False
        self.gross_salary = (self.basic_salary + self.travel_allowance +
                             self.bonus + self.others)
        self.net_salary = self.gross_salary - self.tax
        return True
```

```
def find_min_max_salaries(employees):
    min_salary_emp = max_salary_emp = None

    for emp in employees:
        if not emp.valid:
            continue
        if min_salary_emp is None or emp.net_salary <
      min_salary_emp.net_salary:
            min_salary_emp = emp
        if max_salary_emp is None or emp.net_salary >
      max_salary_emp.net_salary:
            max_salary_emp = emp

    return min_salary_emp, max_salary_emp


def divide_and_conquer(employees, start, end):
    if start == end:
        return employees[start], employees[start]

    mid = (start + end) // 2
    left_min, left_max = divide_and_conquer(employees, start, mid)
    right_min, right_max = divide_and_conquer(employees, mid + 1, end)

    min_salary_emp, max_salary_emp = find_min_max_salaries([left_min,
      left_max, right_min, right_max])
```

```
    return min_salary_emp, max_salary_emp


def main():
    employees = []
    file_path = 'employees_data_.csv'


    try:
        # Read CSV file
        with open(file_path, mode='r') as file:
            reader = csv.DictReader(file)
            for row in reader:
                try:
                    emp = Employee(
                        emp_id=row['Employee ID'],
                        basic_salary=row['Basic Salary'],
                        tax=row['Tax'],
                        travel_allowance=row['Travel Allowance'],
                        bonus=row['Bonus'],
                        others=row['Others']
                    )
                    employees.append(emp)
                except ValueError as e:
                    print(f"Skipping row due to error: {e}")
```

```
        if not employees:
            print("No valid employee data found.")
            return

        # Apply Divide and Conquer
        min_salary_emp, max_salary_emp = divide_and_conquer(employees, 0,
    len(employees) - 1)

        # Output results
        if min_salary_emp and max_salary_emp:
            print(f"Employee with Minimum Net Salary:\nID:
    {min_salary_emp.id}, Net Salary: {min_salary_emp.net_salary}")
            print(f"Employee with Maximum Net Salary:\nID:
    {max_salary_emp.id}, Net Salary: {max_salary_emp.net_salary}")
        else:
            print("No valid employees with positive basic salary found.")

    except FileNotFoundError:
        print(f"Error: The file '{file_path}' was not found.")
    except Exception as e:
        print(f"An error occurred: {e}")

if __name__ == "__main__":
    main()
```

TEST CASES:



Positive Test Cases

| | Expected Output Min | Max |
|---|---|---|
| 1) [250000, 50000, 30000 ...] | 30000 | 250000 |
| 2) [10000, 10000, 10000 ...] | 10000 | 10000 |
| 3) [60000, 125000, 40000 ...] | 40000 | 125000 |
| 4) [99999, 11111, 55555 ...] | 11111 | 99999 |

DATE: / /

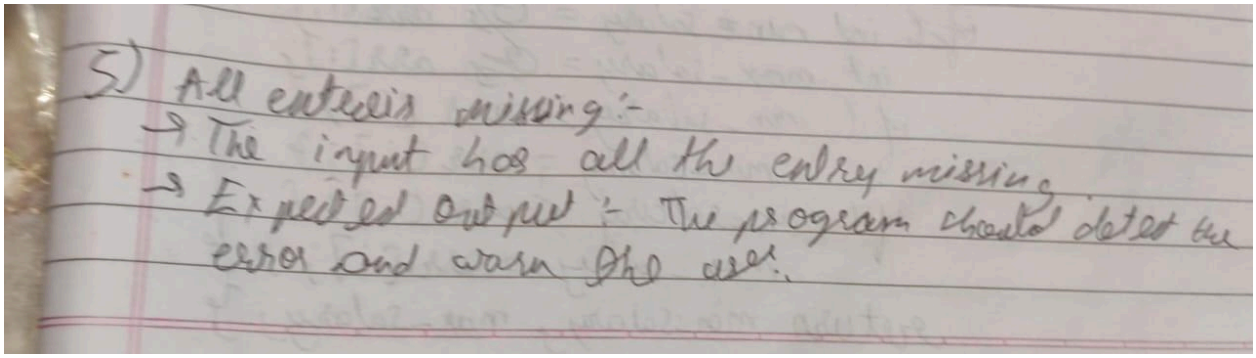
Negative Test Cases

1) Missing or Null Values
→ some of the salary enteries are missing or null.
→ Expected Output:- The program will consider the null or missing no. as default value of '0' and carryon the process.

2) Negative salary components:
→ There could be some salaries which are negative (can not be present).
→ Expected Output:- The program will detect the '-ve' value and warn the user and end the program.

3) Negative Tax components.
→ There could be some tax or other entries as negative, which would incorrectly increase the net salary
→ Expected Output:- The program should warn the user and replace the neg. value with 0 to carry on the code.

4) The salary compone
to Input in String
→ The salary components are given in String data type.
→ Expected Output:- The program should detect the data type mismatch and raise an error.




OUTPUTS FOR THE TEST CASES:

TEST CASE 1:

```
(myenv) PS D:\Dhruv\VJTI\DAA LAB> python3 salary.py
Employee with Minimum Net Salary:
ID: 615, Net Salary: 10016.0
Employee with Maximum Net Salary:
ID: 577, Net Salary: 138177.0
(myenv) PS D:\Dhruv\VJTI\DAA LAB>
```

TEST CASE 2:

```
(myenv) PS D:\Dhruv\VJTI\DAA LAB> python3 salary.py
Employee with Minimum Net Salary:
ID: 3451, Net Salary: 10843.0
Employee with Maximum Net Salary:
ID: 4899, Net Salary: 137989.0
(myenv) PS D:\Dhruv\VJTI\DAA LAB>
```

TEST CASE 3:

```
(myenv) PS D:\Dhruv\VJTI\DAA LAB> python3 salary.py
Employee with Minimum Net Salary:
ID: 3995, Net Salary: 9279.0
Employee with Maximum Net Salary:
ID: 2920, Net Salary: 132979.0
(myenv) PS D:\Dhruv\VJTI\DAA LAB>
```

TEST CASE 4:

```
(myenv) PS D:\Dhruv\VJTI\DAA LAB> python3 salary.py
Employee with Minimum Net Salary:
ID: 227, Net Salary: 6427.0
Employee with Maximum Net Salary:
ID: 2482, Net Salary: 133263.0
(myenv) PS D:\Dhruv\VJTI\DAA LAB>
```

TEST CASE 5:

```
(myenv) PS D:\Dhruv\VJTI\DAA LAB> python3 salary.py
Employee with Minimum Net Salary:
ID: 4726, Net Salary: 9860.0
Employee with Maximum Net Salary:
ID: 14, Net Salary: 134406.0
(myenv) PS D:\Dhruv\VJTI\DAA LAB>
```

TEST CASE 6:

```
Missing value for Basic Salary. Defaulting to 0.
Skipping Employee ID 1986 due to invalid Basic Salary (0.0).
Missing value for Basic Salary. Defaulting to 0.
Skipping Employee ID 1990 due to invalid Basic Salary (0.0).
Missing value for Basic Salary. Defaulting to 0.
Skipping Employee ID 1997 due to invalid Basic Salary (0.0).
An error occurred: 'NoneType' object has no attribute 'valid'
(myenv) PS D:\Dhruv\VJTI\DAA LAB>
```

TEST CASE 7:

```
Skipping Employee ID 1985 due to invalid Basic Salary (-18713.0).
Skipping Employee ID 1993 due to invalid Basic Salary (-12787.0).
Skipping Employee ID 1994 due to invalid Basic Salary (-5059.0).
Skipping Employee ID 1998 due to invalid Basic Salary (-18954.0).
Skipping Employee ID 1999 due to invalid Basic Salary (-5091.0).
Warning: Negative tax value detected for Employee ID 2000. Tax has been set to 0.
An error occurred: 'NoneType' object has no attribute 'valid'
(myenv) PS D:\Dhruv\VJTI\DAA LAB>
```

TEST CASE 8:

```
Warning: Negative tax value detected for Employee ID 1966. Tax has been set to 0.
Warning: Negative tax value detected for Employee ID 1972. Tax has been set to 0.
Warning: Negative tax value detected for Employee ID 1973. Tax has been set to 0.
Warning: Negative tax value detected for Employee ID 1974. Tax has been set to 0.
Warning: Negative tax value detected for Employee ID 2000. Tax has been set to 0.
Employee with Minimum Net Salary:
ID: 1575, Net Salary: 14150.0
Employee with Maximum Net Salary:
ID: 856, Net Salary: 140135.0
(myenv) PS D:\Dhruv\VJTI\DAA LAB>
```

TEST CASE 9:

```
(myenv) PS D:\Dhruv\VJTI\DAA LAB> python3 salary.py
Skipping row due to error: Datatype mismatch detected: Basic Salary should be a float, but got 'one lakh twenty thousand three undred
 and nighteen'.
Employee with Minimum Net Salary:
ID: 394, Net Salary: 12324.0
Employee with Maximum Net Salary:
ID: 264, Net Salary: 136993.0
(myenv) PS D:\Dhruv\VJTI\DAA LAB>
```



TEST CASE 10:

```
(myenv) PS D:\Dhruv\VJTI\DAA LAB> python3 salary.py
No valid employee data found.
(myenv) PS D:\Dhruv\VJTI\DAA LAB>
```

CONCLUSION:

We have used the Divide & Conquer Algorithm to Find the Min Max Salary which has a time complexity of O(n) and found to find gross and net salary of employees.ABC co. ltd. has 2000 employees. I have ued csv file as input for 2000 employees to store and written the program in python using python 8 coding style