Linear Code:

```
import csv

class EmployeeSalary:
    def __init__(self, employee_id,
basic_salary):
        self.employee_id = employee_id
        self.basic_salary = basic_salary
        self.hra = self.calculate_hra()
        self.other_allowances =
self.calculate_other_allowances()
        self.tax = self.calculate_tax()
        self.ppf = self.calculate_ppf()

    def calculate_hra(self):
        return 0.20 * self.basic_salary

    def calculate_other_allowances(self):
        return 0.05 * self.basic_salary

    def calculate_tax(self):
        return 0.25 * self.basic_salary

    def calculate_ppf(self):
        return 0.075 * self.basic_salary

    def calculate_gross_salary(self):
        gross_salary = self.basic_salary +
self.hra + self.other_allowances
        return gross_salary
```

```
    def calculate_net_salary(self):
        gross_salary =
self.calculate_gross_salary()
        net_salary = gross_salary - (self.tax +
self.ppf)
        return net_salary

def process_employee_data(csv_filename):
    employees = []

    try:
        with open(csv_filename, mode='r') as
file:
            csv_reader = csv.DictReader(file)

            for row in csv_reader:
                employee_id = row['EmployeeID']
                basic_salary =
float(row['BasicSalary'])

                employee =
EmployeeSalary(employee_id, basic_salary)

                employees.append(employee)

        for emp in employees:
            gross_salary =
emp.calculate_gross_salary()
            net_salary =
emp.calculate_net_salary()
```

```
            print(f"Employee ID:
{emp.employee_id}, Gross Salary:
{gross_salary:.2f}, Net Salary:
{net_salary:.2f}")

        find_max_net_salary(employees)
        find_min_net_salary(employees)

    except ValueError:
        print("Error: Invalid data format in
CSV. Please ensure all salary fields are
filled.")

def find_max_net_salary(employees):
    max_net_salary = 0
    max_salary_employee = None

    for emp in employees:
        net_salary = emp.calculate_net_salary()
        if net_salary > max_net_salary:
            max_net_salary = net_salary
            max_salary_employee = emp

    if max_salary_employee:
        print(f"\nEmployee with Maximum Net
Salary:")
        print(f"Employee ID:
{max_salary_employee.employee_id}, Net Salary:
{max_net_salary:.2f}")

def find_min_net_salary(employees):
```

```
    min_net_salary = float('inf')
    min_salary_employee = None

    for emp in employees:
        net_salary = emp.calculate_net_salary()
        if net_salary < min_net_salary:
            min_net_salary = net_salary
            min_salary_employee = emp

    if min_salary_employee:
        print(f"\nEmployee with Minimum Net
Salary:")
        print(f"Employee ID:
{min_salary_employee.employee_id}, Net Salary:
{min_net_salary:.2f}")

csv_filename = 'T5.csv'
process_employee_data(csv_filename)
```

Divide And Conquer Algorithm:

```
import csv

class EmployeeSalary:
    def __init__(self, employee_id,
basic_salary):
        self.employee_id = employee_id
        self.basic_salary = basic_salary
        self.hra = self.calculate_hra()
```

```
        self.other_allowances =
self.calculate_other_allowances()
        self.tax = self.calculate_tax()
        self.ppf = self.calculate_ppf()

    def calculate_hra(self):
        return 0.20 * self.basic_salary

    def calculate_other_allowances(self):
        return 0.05 * self.basic_salary

    def calculate_tax(self):
        return 0.25 * self.basic_salary

    def calculate_ppf(self):
        return 0.075 * self.basic_salary

    def calculate_gross_salary(self):
        return self.basic_salary + self.hra +
self.other_allowances

    def calculate_net_salary(self):
        gross_salary =
self.calculate_gross_salary()
        return gross_salary - (self.tax +
self.ppf)

def find_max(a, p, q):
    if p > q:
        return -1, None
    mid = (p + q) // 2
```

```
    if p == q:
        return a[p][1], a[p][0]
    elif q - p == 1:
        if a[p][1] > a[q][1]:
            return a[p][1], a[p][0]
        else:
            return a[q][1], a[q][0]
    else:
        left_max = find_max(a, p, mid - 1)
        right_max = find_max(a, mid + 1, q)
        max_salary = max(left_max[0],
right_max[0])
        if left_max[0] == max_salary:
            max_id = left_max[1]
        else:
            max_id = right_max[1]
        return max_salary, max_id

def find_min(a, p, q):
    if p > q:
        return float('inf'), None
    mid = (p + q) // 2
    if p == q:
        return a[p][1], a[p][0]
    elif q - p == 1:
        if a[p][1] < a[q][1]:
            return a[p][1], a[p][0]
        else:
            return a[q][1], a[q][0]
    else:
        left_min = find_min(a, p, mid - 1)
```

```
        right_min = find_min(a, mid + 1, q)
        min_salary = min(left_min[0],
right_min[0])
        if left_min[0] == min_salary:
            min_id = left_min[1]
        else:
            min_id = right_min[1]
        return min_salary, min_id

def process_employee_data(csv_filename):
    employees = []

    try:
        with open(csv_filename, mode='r') as
file:
            csv_reader = csv.DictReader(file)

            for row in csv_reader:
                employee_id = row['EmployeeID']
                basic_salary =
float(row['BasicSalary'])

                employee =
EmployeeSalary(employee_id, basic_salary)
                employees.append(employee)

        net_salaries = []
        for emp in employees:
            gross_salary =
emp.calculate_gross_salary()
```

```
            net_salary =
emp.calculate_net_salary()
            net_salaries.append((emp.employee_id
, net_salary))
            print(f"Employee ID:
{emp.employee_id}, Gross Salary:
{gross_salary:.2f}, Net Salary:
{net_salary:.2f}")

        if net_salaries:
            max_salary, max_salary_id =
find_max(net_salaries, 0, len(net_salaries) - 1)
            min_salary, min_salary_id =
find_min(net_salaries, 0, len(net_salaries) - 1)
            print(f"Maximum Net Salary:
{max_salary:.2f} (Employee ID:
{max_salary_id})")
            print(f"Minimum Net Salary:
{min_salary:.2f} (Employee ID:
{min_salary_id})")

    except FileNotFoundError:
        print(f"Error: The file {csv_filename}
was not found.")
    except ValueError:
        print("Error: Invalid data format in
CSV. Please ensure all salary fields are
numeric.")

csv_filename = 'T4.csv'
process_employee_data(csv_filename)
```