Algorithms for calculating gross and net salary.

```
Class LoadSalaryData
    Method __init__( dataFrame)
      // Extract basic salary from dataFrame
      self.basic_salary ← dataFrame[column 1]
      // calculate house rent allowance
      self.house_rent ← self.basic_salary * 0.5
      // Calculate Provident fund
      self.provident_fund ← self.basic_salary * 0.12

Class Salary Calculator
    Method __init__( data ) // object of dataFrame
      // Initialize basic salary, house rent
       allowance and provident fund.
       self.basic_salary ← data.basic_salary
       self.house_rent ← data.house_rent
       self.provident_fund ← data.provident_fund.

    Method find_gross ( index)
       self.gross_salary ← self.basic_salary[index]
                            + self.house_rent [index]
       return self.gross_salary

    Method find_net ( index)
       if self.gross_salary < 30000 do:
          self.income_tax ← self.gross_salary * 0.05
       else if self.gross_salary < 41000 do:
          self.income_tax ← self.gross_salary * 0.100
```

P.T.O
→

```
Else do :
    self. income_ tax ← self. gross_salary * 0.15

self. net_salary ← self. gross_salary
                  - self. income_ tax
                  - self. provident_fund[ index]
return self. net_ salary
```

Algorithm for finding minimum and maximum.

```
min_max_iterative ( array):
// inputs: The array containing elements to be
           searched on.
// outputs: Minimum value and it's index and
            Maximum value and it's index.

    int_min  ←  array [0] ;  low ← 0
    int_max  ←  array [0] ;  high ← 0
    for  i = 0  to  length of array do:
        if  array [i]  <=  int_min do:
              int_min ← array [i]
              low ← i
        else if  array [i]  >=  int_max do:
              int_max ← array [i]
              high ← i

    return  low, high, int_min, int_max
```

```
min_max-recursive ( array, low, high):
// inputs : The array, the left pointer
   low and right pointer high.
// outputs : The minimum and it's location
   and maximum and it's location.

    if low == high do:
        return array[low], array[high],
               low, high

    if high == low+1 do:
        if array[high] > array[low] do:
            return array[low], array[high],
                   low, high
        else do:
            return array[high], array[low],
                   high, low

    mid ← (low + high) / 2

lmin, lmax, llow, lhigh ← min_max_recursive(
                          low, mid)
rmin, rmax, rlow, rhigh ← min_max-recursive(
                          mid+1, high)

    max-val ← max( lmax, rmax)
    min-val ← min( lmin, rmin)
```

P.T.O
→

```
if max-val == rmax do:
    high ← rhigh
else do:
    high ← lhigh

if min-val == rmin do:
    low ← rlow
else do:
    low ← llow

return min-val, max-val, low, high
```

Test cases :

Positive test - cases :

when basic salary has all values and all are non - negative. it will provide the correct ans.

expected output : the min and max of salaries and their positions as we

Negative test cases :

1] Basic salary columns has missing data :-

the program will check for that and return an error :

expected output : " Error : value not present "

2] Basic salary has negative data :

Expected output : " Error : negative values present "

Time complenities of the following are:

1] For iterative function:

The for loop runs for n values from 1 to n, each itearation has four basic operations. One for return.

$$\therefore \quad T(n) = 1 + \sum_{i=1}^{n} 4$$

$$= 4n + 1$$

$\therefore$ ~~$O(n)$~~ = $\therefore$ $T(n) = O(n)$

Hence function takes linear time.

2] For the recursive function:

The recursive function works on the following equation:

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + O(1)$$

where $T\left(\frac{n}{2}\right)$ is for both left and right halves and $O(1)$ is for the comparisons.

$$\therefore \quad T(n) = 2T\left(\frac{n}{2}\right) + O(1)$$

By master's theorem,

$a = 2$, $b = 2$ and $d = 0$

$\therefore \theta(n^0) = \theta(1)$.

$\therefore \quad a > b^d \quad \ldots \{2^1 > 2^0\}$

then $T(n) = \cancel{\theta n^d} \; O(n^{\log_b a})$

$\therefore \quad T(n) = O(n^{\log_2 2})$

$= O(n)$

Hence this function is also linear.

Conclusion :

This problem was solved using both linear approach and divide-and-conquer approach. However, both resulted in $O(n)$ time complexity. Signifying the fact that not all problems can be solved with better time complexity if divide and conquer is applied.

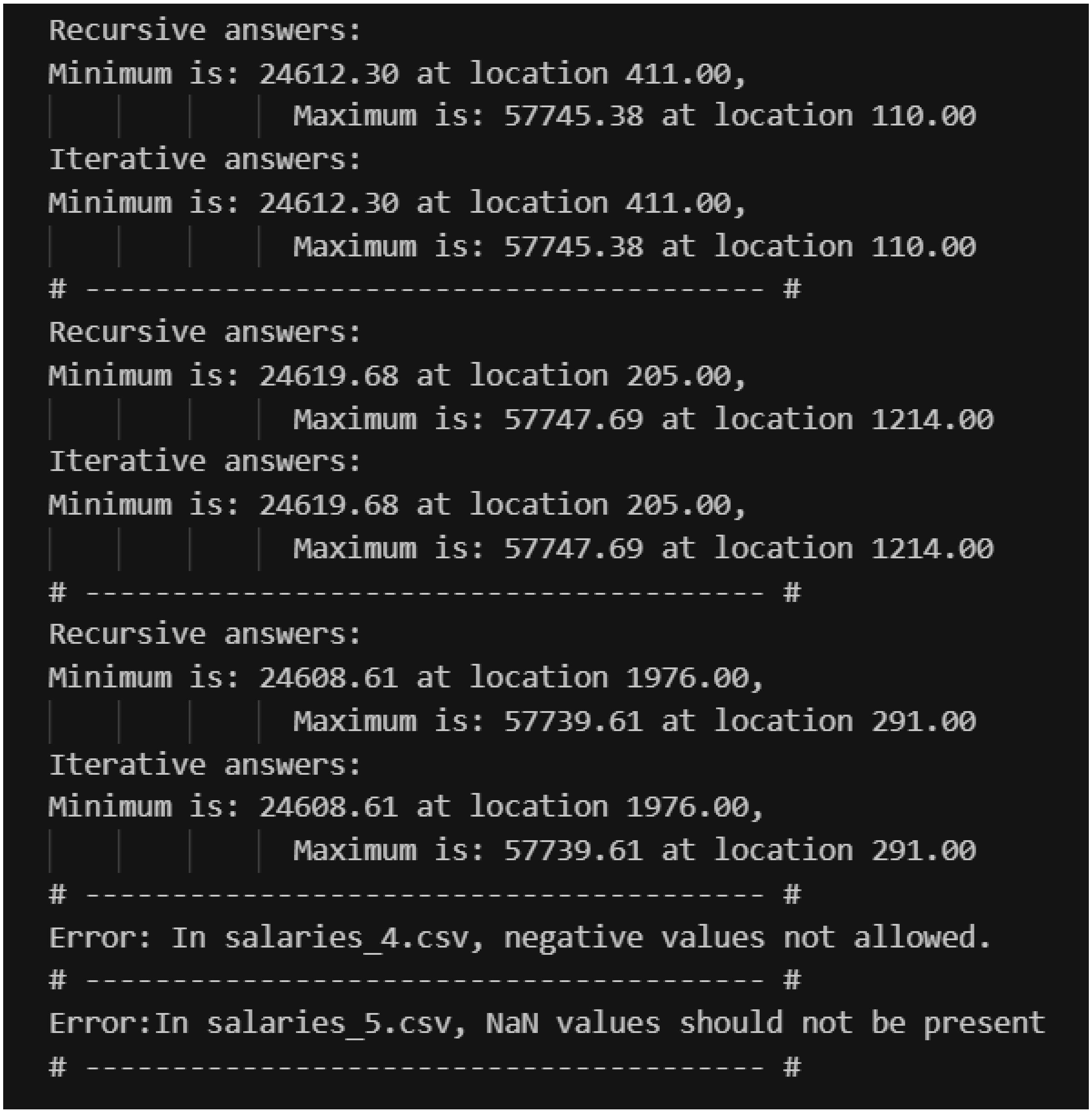
Code:

Python

```python
"""
Author: Ayush Bothra
Date: 20-08-2024
Aim: Code for finding the maximum element in an unsorted array
using both iterative and divide and conquer approach
and get a solution in O(n) time.
"""

# Importing the necessary libraries
import pandas as pd
import numpy as np


class LoadSalaryData():
    def __init__(self, dataFrame):
    # extract data from the dataframe taken:
        self.df = pd.read_csv(dataFrame)
        self.basic_salary = self.df[self.df.columns[1]]
        self.house_rent = self.basic_salary * 0.5
        self.provident_fund = self.basic_salary * 0.12


class SalaryCalculator:
    def __init__(self, data):
        self.basic_salary = data.basic_salary
        self.house_rent = data.house_rent
        self.provident_fund = data.provident_fund

    def find_gross(self, index):
        self.gross_salary = self.basic_salary[index] + self.house_rent[index]
        return self.gross_salary

    def find_net(self, index):
        if self.gross_salary < 30000:
            self.income_tax = gross_salary * 0.05
        elif self.gross_salary < 41000:
            self.income_tax = self.gross_salary * 0.1
        else:
            self.income_tax = gross_salary * 0.15
        self.net_salary = self.gross_salary - self.income_tax -
self.provident_fund[index]
```

```
        return self.net_salary


class GetMinMax:
    def __init__(self, array):
        self.array = array

    def min_max_iterative(self):
        if len(self.array) == 0:
            return None, None, None, None
        if len(self.array) == 1:
            return 1, 1, self.array[0], self.array[0]
        int_min = self.array[0]
        int_max = self.array[0]
        low, high = 0, 0

        # Check for both maximum and minimum in a single iteration
        for i in range(len(self.array)):
            if self.array[i] <= int_min:
                int_min = self.array[i]
                low = i
            elif self.array[i] >= int_max:
                int_max = self.array[i]
                high = i

        return int_min, int_max, low, high

    def min_max_recursive(self, low, high):
        if len(self.array) == 0:
            return None, None, None, None
        # If there's only one element in the divided array
        if low == high:
            return self.array[low], self.array[high], low, high
        # If there are two elements in the divided array
        if high == low + 1:
            if self.array[high] > self.array[low]:
                return self.array[low], self.array[high], low, high
            else:
                return self.array[high], self.array[low], high, low

        # Compute the mid of the array
        mid = (low + high) // 2

        # Recursively traverse the right and left arrays while dividing them
        lmin, lmax, llow, lhigh = self.min_max_recursive(low, mid)
        rmin, rmax, rlow, rhigh = self.min_max_recursive(mid + 1, high)

        max_val = max(lmax, rmax)
        min_val = min(lmin, rmin)

        if max_val == rmax:
            high = rhigh
        else:
            high = lhigh

        if min_val == rmin:
            low = rlow
        else:
            low = llow

        return min_val, max_val, low, high


if __name__ == "__main__":
    df_storer = []

    for i in range(1, 6):
        df = (f'salaries_{i}.csv')
        df_storer.append(df)

    for df in df_storer:
        #  checking for negative values or empty array values first
        data = LoadSalaryData(df)
        if np.any(data.basic_salary < 0):
            print(f"Error: In {df}, negative values not allowed.")
            print('# --------------------------------------- #')
            continue
        elif np.any(np.isnan(data.basic_salary)):
            print(f"Error:In {df}, NaN values should not be present")
            print('# --------------------------------------- #')
            continue

        calculate_salary = SalaryCalculator(data)
        gross_salary_all = []
        net_salary_all = []

        for i in range(2000):
            gross_salary = calculate_salary.find_gross(i)
            gross_salary_all.append(gross_salary)
            net_salary = calculate_salary.find_net(i)
            net_salary_all.append(net_salary)

        min_max = GetMinMax(net_salary_all)

        print('Recursive answers:')

        min_recursive, max_recursive, recursive_low, recursive_high =
min_max.min_max_recursive(0, len(net_salary_all) - 1)
        print(f'''Minimum is: {min_recursive:.2f} at location
{recursive_low:.2f},
              Maximum is: {max_recursive:.2f} at location
{recursive_high:.2f}''')

        print('Iterative answers:')

        min_iterative, max_iterative, iterative_low, iterative_high =
min_max.min_max_iterative()
        print(f'''Minimum is: {min_iterative:.2f} at location
{iterative_low:.2f},
              Maximum is: {max_iterative:.2f} at location
{iterative_high:.2f}''')

        print('# --------------------------------------- #')
```

Output:



```
Recursive answers:
Minimum is: 24612.30 at location 411.00,
              Maximum is: 57745.38 at location 110.00
Iterative answers:
Minimum is: 24612.30 at location 411.00,
              Maximum is: 57745.38 at location 110.00
# -------------------------------------- #
Recursive answers:
Minimum is: 24619.68 at location 205.00,
              Maximum is: 57747.69 at location 1214.00
Iterative answers:
Minimum is: 24619.68 at location 205.00,
              Maximum is: 57747.69 at location 1214.00
# -------------------------------------- #
Recursive answers:
Minimum is: 24608.61 at location 1976.00,
              Maximum is: 57739.61 at location 291.00
Iterative answers:
Minimum is: 24608.61 at location 1976.00,
              Maximum is: 57739.61 at location 291.00
# -------------------------------------- #
Error: In salaries_4.csv, negative values not allowed.
# -------------------------------------- #
Error:In salaries_5.csv, NaN values should not be present
# -------------------------------------- #
```