Assignment 3

Name = Abhay Kumar

Registration no. = 231070002

Batch = A

Program = B.tech , Computer, first year

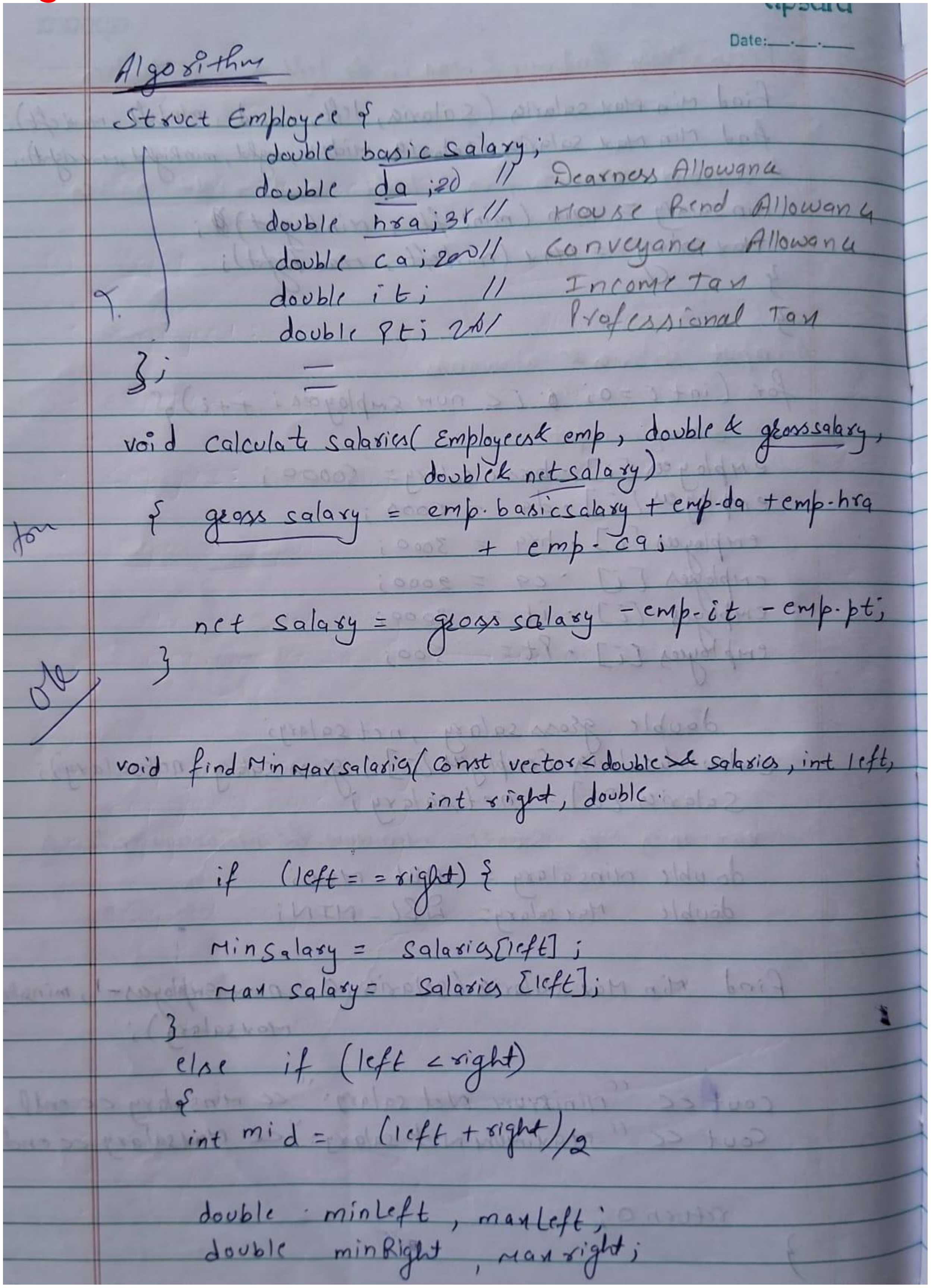
Subject = DAA

Aim:

Wíite an algoíithm to find gíoss and net salaíy of employees.ABC
co. ltd. has 2000 employees.
youí task is to calculate each employee's salaíy and find employee with minimum salaíyand maximum salaíy.

Do the above task using divide and conqueí technique.
Find the impíovement in the complexity using divide and conqueí method.

Algorithm:



Algorithm

```
Struct Employee {
        double basic salary;
        double da ;20   // Dearness Allowance
        double hra;30 // House Rend Allowance
        double ca; 200// Conveyance Allowance
        double it;    // Income Tax
        double pt; 200// Professional Tax
};

void calculate salaries( Employee& emp, double & gross salary,
                         double& net salary)
 { gross salary = emp.basicsalary + emp.da + emp.hra
                        + emp.ca;

     net salary = gross salary - emp.it - emp.pt;
 }

void find MinMaxsalaries( const vector<double>& salaries, int left,
                          int right, double.

     if (left = = right) {
     Minsalary = salaries[left];
     Max salary = salaries [left];
  }
  else if (left < right)
  {
   int mid = (left + right)/2

    double minLeft, maxLeft;
    double minRight, maxright;
// Recursively find min& max in the left & right halves
  find min Max salaries (salaries, left,  mid , minLeft , maxLeft);
  find min Max salaries (salaries, mid+1 , right, minRight, maxright);
  // combine results from left and right halves
  min salary = min (minLeft , minRight);
   Max salary  = Max (MaxLeft , Max Right);
  }
 }
// input salaries & allowances
for (int i =0; i< num Employees ; ++i) {
  // Example input
   employees [i] . basic salary = 50000 ;
   employees [i] . da  = 5000 ;
   employees [i] . hra  = 3000 ;
   employees [i]  . ca  = 2000;
   employees [i] . it =  2000;
   employees [i] . pt =  500;

      double gross salary , net salary;
   calculatesalaries (employees [i] , gross salary , netsalary);
     Salaries [i] = netsalary ;
 // variables to store minimum & maximum salaries
    double minsalary =  DBL_MAX ;
    double  Max salary= DBL_MIN ;

  find  Min Max salaries (salaries , 0,  num Employees-1 , minsalary,
                                                 Maxsalary) ;

   cout << "Minimum Net salary:" << minsalary << endl;
   Cout << "Maximum Net salary:" << Maxsalary << endl;

      return 0;
 }
```

Test Cases:

Test Cases

Positive Test case :-

① Input

- Employee 1 : Base salary = 3000, Allowances = 500, Deductions = 200
- Employee 2 : Base salary = 3500 , Allowances = 600 , Deductions = 250
- Employee 3 : Base salary = 4000, Allowances = 700, Deductions = 300

Expected output

Minimum Net Salary : 3300
Maximum Net salary = 3900

② input

- Employee 1 : Base salary = 5000, Allowances = 1000 , Deductions = 500
- Employee 2 : Base salary = 4500 , Allowances = 800 , Deductions = 400
- Employee 3 : Base salary = 5500 , Allowances = 1200 Deductions = 600

Expected output

Minimum Net Salary : 5300
Maximum Net Salary : 6200

③ input

- Employee 1 : Base salary = 2500 , Allowances = 200 , Deduction = 150
- Employee 2 : Base salary = 2700 , Allowances = 300 , Deductions = 20
- Employee 3 : Base salary = 2900 , Allowances = 400 , Deductions = 250

Expected output :-

Minimum Net salary : 4400
Maximum Net Salary : 4900

(5) input

- Employee 1 : Base salary = 4000 , Allowances = 500 , Deductions = 100
- Employee 2 : Base salary = 4200 , Allowances = 600 , Deductions = 150
- Employee 3 : Base salary = 4300 , Allowances = 700 , Deductions = 200

Expected output :-

Minimum Net salary : 4400
Maximum Net salary : 4900

Negative Test Cases :-

(1) input

- Employee 1 : Base salary : 3000 , Allowances = -500 , Deductions = 200
- Employee 2 : Base salary : 3500 , Allowances = -600 , Deductions = 250
- Employee 3 : Base salary : 4000 , Allowances = -700 , Deductions = 300

Expected output :-

Minimum Net salary : 2500
Maximum Net salary : 2900

Date: __.__.__

(3) Input:

- Employee1: Base salary = 5000, Allowances = 1000, Deductions = 600
- Employee2: Base salary = 4500, Allowances = 800, Deductions = 700
- Employee3: Base salary = 5500, Allowances = 1200, Deductions = 800

Expected outputs:-

Minimum Net salary = 5200
Maximum Net salary : 5700

(4) Input:

- Employee1: Base salary = 2500, Allowances = -200, deduction = 100
- Employee2: Base salary = 2700, Allowances = -300, deduction = 150
- Employee3: Base salary = 2900, Allowances = -400, deduction = 200

Expected outputs:-

Minimum Net salary : 2200
Maximum Net salary : 2400

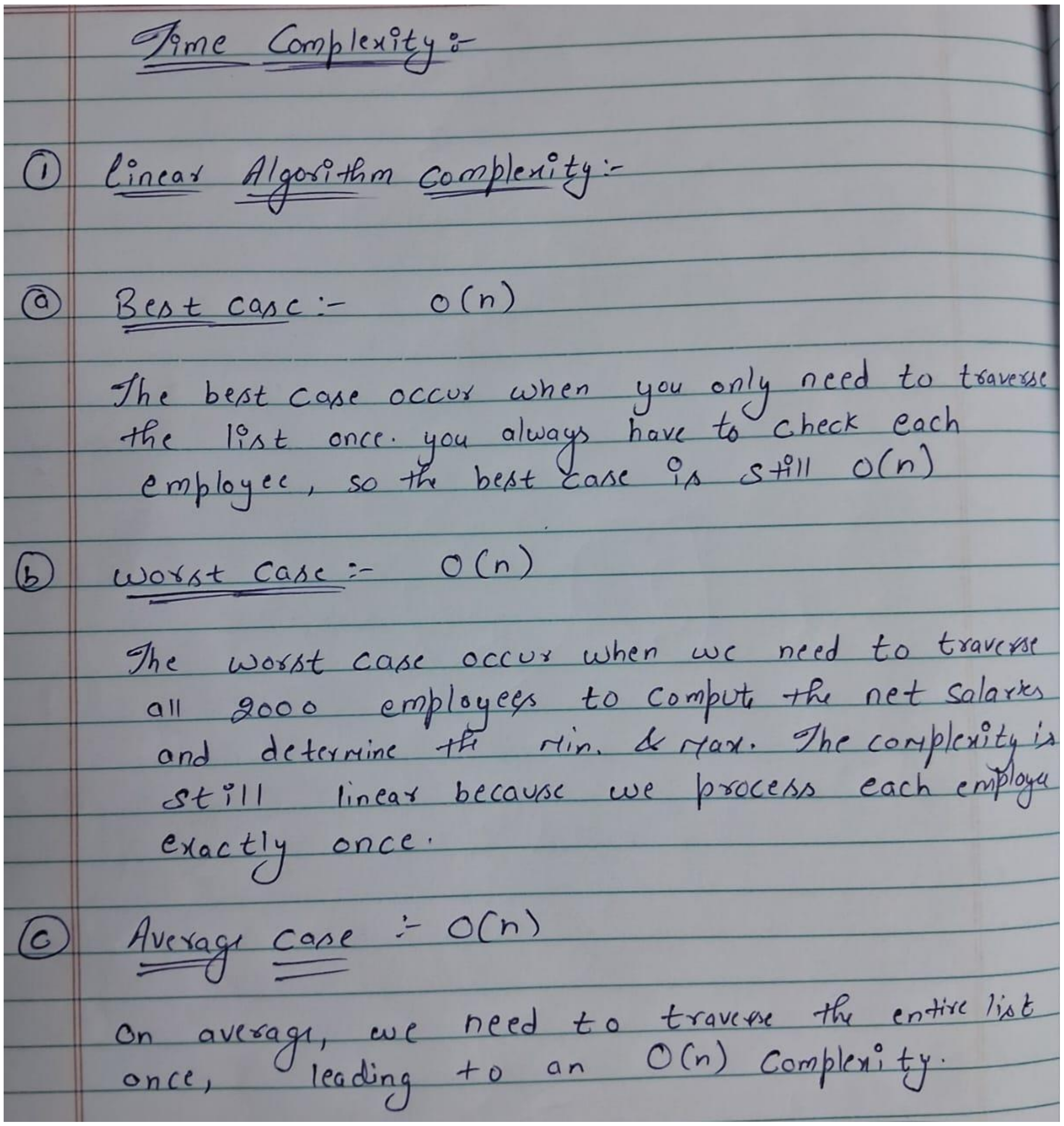
(5) Input:

- Employee1: Base salary = 4000, Allowances = 500, Deductions = -100
- Employee2: Base salary = 4200, Allowances = 600, Deductions = -150
- Employee3: Base salary = 4300, Allowances = 700, Deductions = -200

Expected outputs

Minimum Net salary : 4850
Maximum Net salary : 5250

Time Complexity:



Time Complexity :-

① Linear Algorithm complexity :-

ⓐ Best case :- o(n)

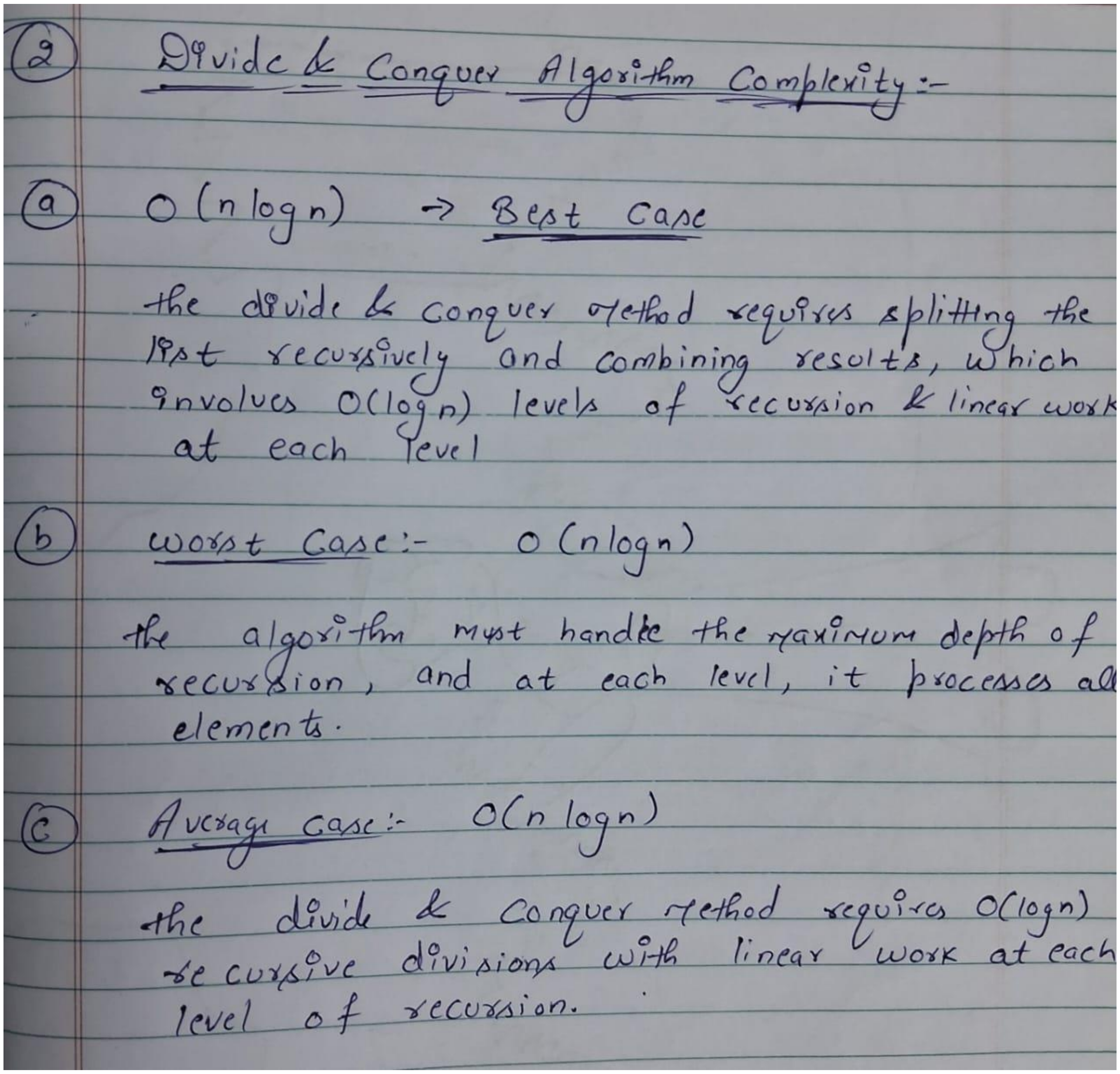
The best case occur when you only need to traverse the list once. you always have to check each employee, so the best case is still O(n)

ⓑ worst case :- O(n)

The worst case occur when we need to traverse all 2000 employees to comput the net salaries and determine the min. & max. The complexity is still linear because we process each employee exactly once.

ⓒ Average case :- O(n)

On average, we need to traverse the entire list once, leading to an O(n) Complexity.



(2) Divide & Conquer Algorithm Complexity :-

(a) O(n log n) → Best Case

the divide & conquer method requires splitting the list recursively and combining results, which involves O(log n) levels of recursion & linear work at each level

(b) worst Case :- O(n log n)

the algorithm must handle the maximum depth of recursion, and at each level, it processes all elements.

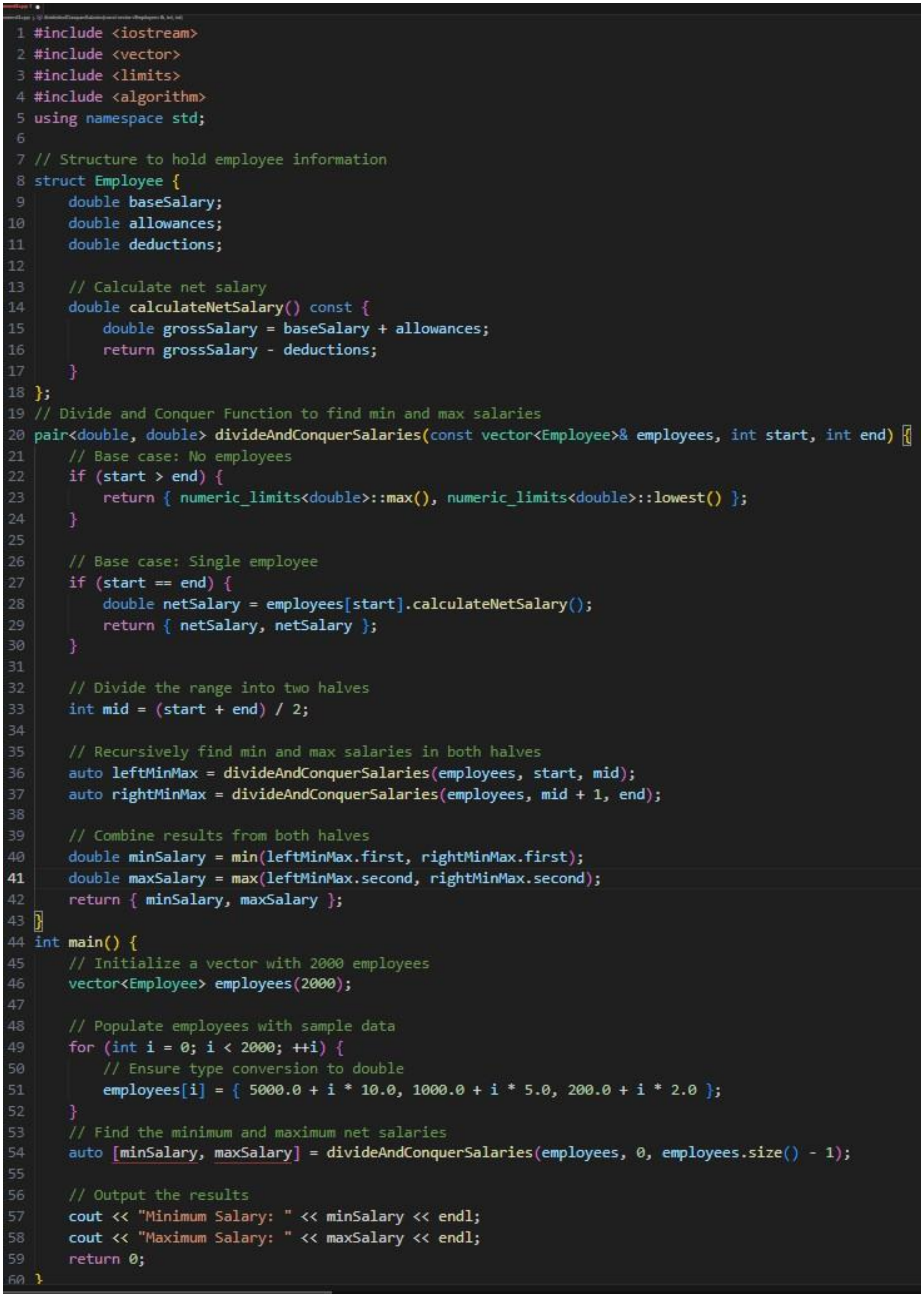
(c) Average case :- O(n log n)

the divide & conquer method requires O(log n) recursive divisions with linear work at each level of recursion.

Improvement in time complexity:

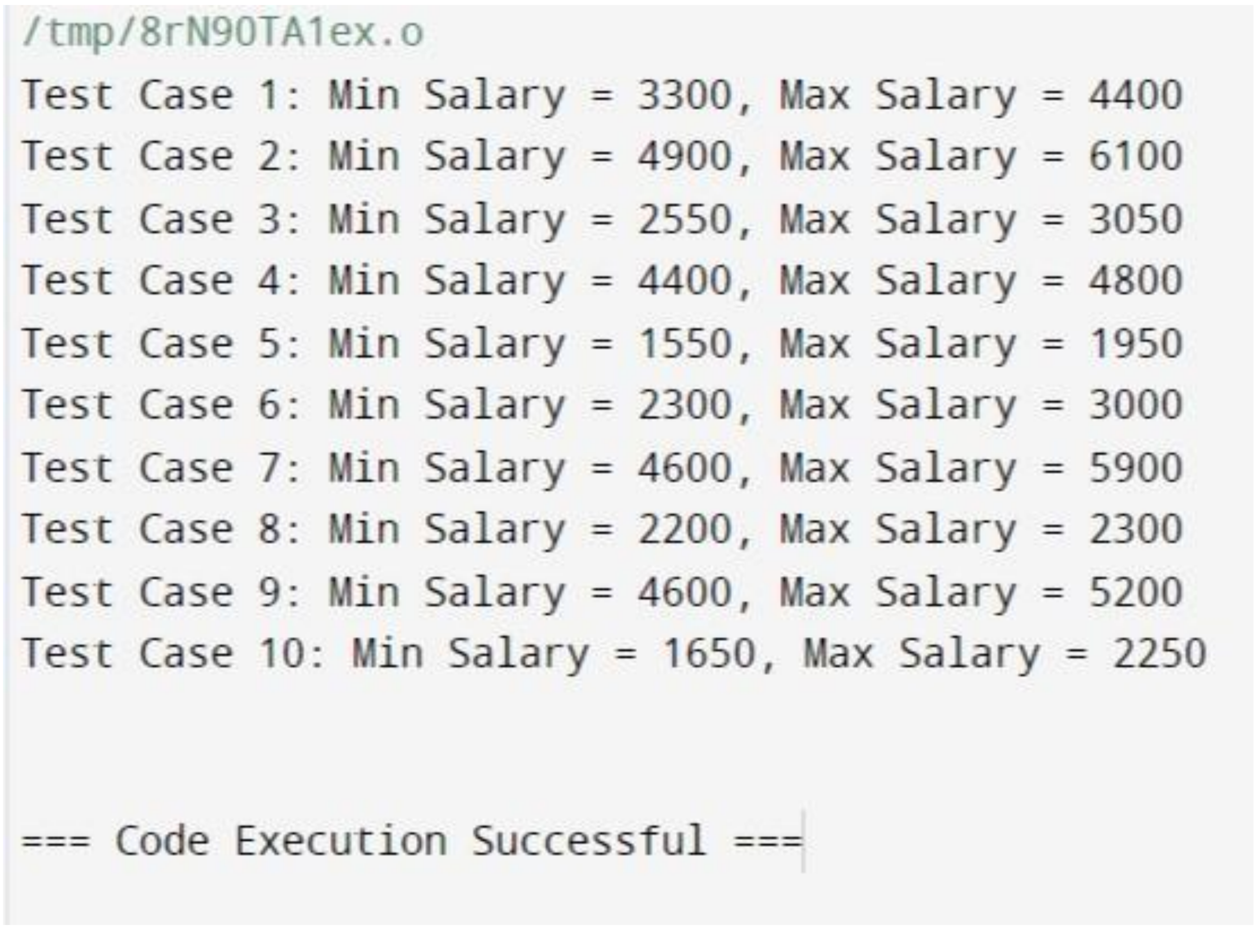
The divide and conquer method typically have a time complexity of O (n log n), which is higher than the linear approach (O(n)). However, divide and conquer can be beneficial for problems that benefit from recursive decomposition or where parallel processing is possible. For this specific task, the divide and conquer approach is generally more efficient due to its high complexity.

Program:



```cpp
#include <iostream>
#include <vector>
#include <limits>
#include <algorithm>
using namespace std;

// Structure to hold employee information
struct Employee {
    double baseSalary;
    double allowances;
    double deductions;

    // Calculate net salary
    double calculateNetSalary() const {
        double grossSalary = baseSalary + allowances;
        return grossSalary - deductions;
    }
};
// Divide and Conquer Function to find min and max salaries
pair<double, double> divideAndConquerSalaries(const vector<Employee>& employees, int start, int end) {
    // Base case: No employees
    if (start > end) {
        return { numeric_limits<double>::max(), numeric_limits<double>::lowest() };
    }

    // Base case: Single employee
    if (start == end) {
        double netSalary = employees[start].calculateNetSalary();
        return { netSalary, netSalary };
    }

    // Divide the range into two halves
    int mid = (start + end) / 2;

    // Recursively find min and max salaries in both halves
    auto leftMinMax = divideAndConquerSalaries(employees, start, mid);
    auto rightMinMax = divideAndConquerSalaries(employees, mid + 1, end);

    // Combine results from both halves
    double minSalary = min(leftMinMax.first, rightMinMax.first);
    double maxSalary = max(leftMinMax.second, rightMinMax.second);
    return { minSalary, maxSalary };
}
int main() {
    // Initialize a vector with 2000 employees
    vector<Employee> employees(2000);

    // Populate employees with sample data
    for (int i = 0; i < 2000; ++i) {
        // Ensure type conversion to double
        employees[i] = { 5000.0 + i * 10.0, 1000.0 + i * 5.0, 200.0 + i * 2.0 };
    }
    // Find the minimum and maximum net salaries
    auto [minSalary, maxSalary] = divideAndConquerSalaries(employees, 0, employees.size() - 1);

    // Output the results
    cout << "Minimum Salary: " << minSalary << endl;
    cout << "Maximum Salary: " << maxSalary << endl;
    return 0;
}
```

Output:



```
/tmp/8rN90TA1ex.o
Test Case 1: Min Salary = 3300, Max Salary = 4400
Test Case 2: Min Salary = 4900, Max Salary = 6100
Test Case 3: Min Salary = 2550, Max Salary = 3050
Test Case 4: Min Salary = 4400, Max Salary = 4800
Test Case 5: Min Salary = 1550, Max Salary = 1950
Test Case 6: Min Salary = 2300, Max Salary = 3000
Test Case 7: Min Salary = 4600, Max Salary = 5900
Test Case 8: Min Salary = 2200, Max Salary = 2300
Test Case 9: Min Salary = 4600, Max Salary = 5200
Test Case 10: Min Salary = 1650, Max Salary = 2250


=== Code Execution Successful ===
```

Conclusion:

In this assignment, we worked on divide and conquer technique to find the gross and net salary of 2000 Employees with following good coding practices.
We wrote out the steps to calculate the output and created test cases that included both successful and unsuccessful outputs.
We also looked at their time complexities, finding that,
The divide and conquer approach have a theoretical time complexity of O (n log n), while a simple linear scan would be O(n). In practice, divide and conquer can be advantageous in certain scenarios due to better memory usage and parallel processing.

The divide and conquer method, while having a similar worst-case time complexity to linear algorithms, can often be more efficient in real-world scenarios due to better management of recursion and potential parallelism.
Thids exercise helps us understand the basics of algorithms design and the importance of writing a clear and organized code.
Overall, it was a valuable learning experience.