Ahaan Desai
2310700016
Batch. A1 (1-20)

DAA Lab 3.

```
Class User:
    initialise (basic_salary, provident_fund):
        this.basic_salary = basic_salary
        this.hra = 0.1 * this.basic_salary
        this.income_tax = 0.1 * this.basic_salary
        if (20000 < this.basic_salary <= 30000)
            this.income_tax = 0.2 * this.basic_salary
        else if (30000 < this.basic_salary)
            this.income_tax = 0.3 *
                this.basic_salary
        this.provident_fund = provident_fund

gross_salary (User u)
    // Input: Object of class User
    // Output: Gross salary of g. user
    gross_salary = u.basic_salary + u.hra
    return gross_salary.

net_salary (User u)
    // Input: Object of class User
    // Output: Net salary of user
    net_salary = gross_salary(u) - u.income_tax
                 - u.provident_fund
    return net_salary.
```

```
gross_salaries (users)
// Input: Array of User objects
// Output: Array of gross salaries
gross_salaries = []
for i: 0 to users.length - 1:
    gross_salaries.append(
        gross_salary(users[i])
    )
return gross_salaries
```

```
net_salaries(users)
// Input: Array of User objects
// Output: Array of net salaries
net_salaries = []
for i=0 to users.length - 1:
    net_salaries.append(
        net_salary(users[i])
    )
return net_salaries
```

DAA Lab 3

Algorithm :-

I) Maximum Algorithm & Minimum Algorithm (Iterative) :-

```
find_max_min_iterative (array)
// Input :- array of salaries
// Output :- maximum & minimum salaries
             (tuple)
max_salary = array[0]
min_salary = array[0]
n = length(array)
for i = 1 to n-1:
    max_salary = max(max_salary, array[i])
    min_salary = min(min_salary, array[i])
return min_salary, max_salary
```

II) Maximum Algorithm (Divide & Conquer)

```
find_max_divconq (array, start, end):
// Input → array of salaries,
           start & end index indices.
// Output → Maximum salary.
if start > end:
    return -1
else if start == end:
    return array[start]
else if end - start + 1 == 2:
    return max(array[start], array[end])
mid = (start + end) // 2
return max( find_max_divconq(array, start, mid),
            find_max_divconq(array, mid+1, end))
```

14) Minimum Algorithm (Divide & Conquer):

```
find_min_divconq (array, start, end)
// Input : array of salaries, start & end indices
// Output : minimum salary.
if start > end:
    return ∞
else if start == end:
    return arr[start]
else if end - start + 1 == 2:
    return min (arr[start], arr[end])
mid = (start + end) // 2
return min ( find_min_divconq (array, start, mid-1), array[mid],
             find_min_divconq (array, mid+1, end))
```

Time Complexity:

A) Linear algorithm

The basic step is updating the minimum or maximum value, which is repeated n times, for all elements in the array.

$$T(n) = 1 + \sum_{i=1}^{n-1} (1)$$

[Initialization of value] [Loop] [Updating value]

$$= 1 + n - 1$$

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

Hence the time complexity is O(n)

B) Divide & conquer algorithm:

The recurrence relation to find the minimum or maximum is

$$T(n) = T(n/2) + T(n/2) + 1$$

$T(n/2)$: searching in left half; $T(n/2)$: searching in right half; 1: calculating value.

$$\therefore T(n) = 2T(n/2) + 1$$

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

Hence, by the master method,

$a = 2, \quad b^d = 2^0 = 1$

$\therefore a > b^d$

Hence, the time complexity should be $O(n^d \log n)$ i.e. $O(n \log n)$