findminmax (arr[], low, high)
// searches for maximum and minimum salary value
// from arr[] by using recursive divide and conquer
// input: an array arr[] and indices low, high
// output: The maximum and minimum value present at that
// particular index
int low = 0
int high = arr.size() - 1
if (low == high) {
  return {arr[low], arr[low]};
}
else if (high == low + 1) {
  if (arr[low] < arr[high]) {
    return {arr[low], arr[high]};
  } else {
    return {arr[high], arr[low]};
  }
}
else {
  int mid = low + (high - low)/2;
  ~~minmax~~ leftResult = findminmax(arr[], low, mid);
  ~~minmax~~ rightResult = findminmax(arr[], mid+1, high);

  minimum = min(leftresult.~~min~~min, rightresult.min);
  maximum = max(leftresult.max, rightresult.max);

  return minimum, maximum;
}

⇒ linear algo

```
fun minmax (arr [ ])
// searches for maximum and minimum salary value from
// given array by using iterative approach i.e by comparing
// every element present in array
// input: an array arr [ ] containing the salaries of employees
// output: maximum and minimum values of the employee's
//          salary
max_salary = arr [0]
min_salary = arr [0]
for (int i = 1; i < arr.size() ; i++) {
  if (a[i] > max_salary) {
      max_salary = a[i] ; }
  if (a[i] < min_salary) {
      min_salary = a[i] ; } }
  return min_salary, max_salary ; }
```

Time complexity using master theorem

$T(n) = 2T\left(\frac{n}{2}\right) + O(1)$ {O(1) since base case we take only 1 element is present in array}

$a = 2$ {since we are using the recursive call of function twice}
$b = 2$ {since we are using divide and conquer recursively each time f^n is called the array is split in half by using divide method}

general formula: $T(n) = aT\left(\frac{n}{b}\right) + f(n)$

$n^k = 1$

$K = 0$

$a > b^k$
$2 > 2^0$
$2 > 1$
} This satisfies one of the master theorem's case

Hence we use this formula: $n^{\log_b a}$

$\log_b a = \log_2 2 = 1$

n^1

n

Thus the time complexity of the algorithm is $O(n)$

for linear approach

base case is only 1 element present in array i.e O(1)

$$\sum_{i=0}^{n} 1 = (n-1) + 1$$

$$= O(n)$$

Date
Page

Conclusion:

In this experiment we have studied how we can find maximum and minimum salary of an employee from using both gross and net salary. This was implemented using linear search but time complexity was found to be O(n) so to improvise it we used divide and conquer recursively and even then there was no improvement in the time complexity as it was O(n). I took a holistic consideration both positive and negative test cases and displayed appropriate messages wherever required as expected output.