DAA Lab Assignment-3

Name : Ashutosh Kumar

Registration: 231070006

Batch :A

Aim :

Write an algorithm to find gross and net salary of employees.
ABC co. ltd. has 2000 employees.
your task is to calculate each employees salary and find employee with minimum salary and maximum salary.

Function to generate random names and salary :

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstdlib>
#include <ctime>
#include <vector>
using namespace std;

string generateRandomName() {
vector<string> firstNames = {"George", "John", "Thomas",
    "Abraham", "Theodore", "Franklin",
    "John", "Ronald", "Barack", "Joe"};
vector<string> lastNames = {"Washington", "Adams", "Jefferson",
    "Lincoln", "Roosevelt", "Roosevelt",
    "Kennedy", "Reagan", "Obama", "Biden"};

    string firstName = firstNames[rand() % firstNames.size()];
    string lastName = lastNames[rand() % lastNames.size()];

    return firstName + " " + lastName;
}

int generateRandomSalary() {
    return rand() % 90001 + 18000; // Random salary between 10,000 and
100,000
}

int main() {
    srand(static_cast<unsigned int>(time(0))); // Seed for random
number generation

    ofstream file("input5.csv");

    if (!file.is_open()) {
        cerr << "Error opening file!" << endl;
        return 1;
    }

    // Write the header
    file << "Name,Salary\n";

    // Generate and write 2000 records
    for (int i = 0; i < 2000; ++i) {
        string name = generateRandomName();
```

```
        int salary = generateRandomSalary();
        file << name << "," << salary << "\n";
    }

    file.close();
    cout << "CSV file created successfully!" << endl;

    return 0;
}
```

Function to create CSV file as an output

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>
using namespace std;


void merge(vector<double>& arr, int left, int mid, int right)
{
    int n1 = mid - left + 1;
    int n2 = right - mid;

    // Create temp vectors
    vector<int> L(n1), R(n2);

    // Copy data to temp vectors L[] and R[]
    for (int i = 0; i < n1; i++)
        L[i] = arr[left + i];
    for (int j = 0; j < n2; j++)
        R[j] = arr[mid + 1 + j];

    int i = 0, j = 0;
    int k = left;

    // Merge the temp vectors back
    // into arr[left..right]
    while (i < n1 && j < n2)
```

```
    {
        if (L[i] <= R[j])
        {
            arr[k] = L[i];
            i++;
        }
        else
        {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    // Copy the remaining elements of L[],
    // if there are any
    while (i < n1)
    {
        arr[k] = L[i];
        i++;
        k++;
    }

    // Copy the remaining elements of R[],
    // if there are any
    while (j < n2)
    {
        arr[k] = R[j];
        j++;
        k++;
    }
}

// begin is for left index and end is right index
// of the sub-array of arr to be sorted
void mergeSort(vector<double> &arr, int left, int right)
{
    if (left >= right)
        return;

    int mid = left + (right - left) / 2;
    mergeSort(arr, left, mid);
    mergeSort(arr, mid + 1, right);
    merge(arr, left, mid, right);
}
```

```cpp
int main()
{
    ifstream inputFile("input5.csv"); // Input CSV file
    ofstream outputFile("output5.csv");              // Output CSV file
    vector<double>v;
        if (!inputFile.is_open() || !outputFile.is_open())
    {
        cout << "Error opening file!" << endl;

    }

    string line;
    // Write the header for the output file
    outputFile << "Name,Salary,Tax,Home Rent,Bonus\n";

    // Skip the header line in the input file
    getline(inputFile, line);

    // Process each line
    while (getline(inputFile, line))
    {
        stringstream ss(line);
        string name, salaryStr;
        getline(ss, name, ',');
        getline(ss, salaryStr, ',');

        double salary = stod(salaryStr);
        v.push_back(salary);
        double tax = 0.30 * salary;
        double homeRent = 0.12 * salary;
        double bonus = 0.15 * salary;

        // Write the results to the output file
        outputFile << name << "," << salary << "," << tax << "," <<
homeRent << "," << bonus << "\n";
    }

    inputFile.close();
    outputFile.close();


    cout << "Calculations completed and output saved to
'output_finances.csv'." << endl;
    mergeSort(v,0,2000);
    if(v[1]<0){
        cout<<"Salary can't be negative";
```

```
    }
    else if(v.empty()){
        cout<<"No Data Present";
    }
    else{
    cout<<"Min Salary -> "<<v[1]<<endl;
    cout<<"Max Salary -> "<<v[v.size()-1];
}
    return 0;
}
```

OutPut:

```
 Calculations completed and output saved to 'output_finances.csv'.

Min Salary -> 10008
Max Salary -> 42755
PS C:\Users\Dell\Desktop\c++>

Calculations completed and output saved to 'output_finances.csv'.
Min Salary -> 10019
Max Salary -> 42715
PS C:\Users\Dell\Desktop\c++>

Error opening file!
Calculations completed and output saved to 'output_finances.csv'.
PS C:\Users\Dell\Desktop\c++>

Calculations completed and output saved to 'output_finances.csv'.
Min Salary -> 10003
Max Salary -> 42702
PS C:\Users\Dell\Desktop\c++>

Calculations completed and output saved to 'output_finances.csv'.
Salary can't be negative
PS C:\Users\Dell\Desktop\c++>
```

Conclusion :

"The provided C++ code demonstrates the implementation of the Merge Sort algorithm, a classic example of the divide-and-conquer strategy. The algorithm works by recursively splitting the array into smaller subarrays, sorting each one, and then merging them back together in sorted order. This approach is efficient, with a time complexity of O(n log n), making it well-suited for sorting large datasets. Additionally, we explored file handling in C++ and applied the divide-and-conquer strategy in a real-time project."

PAGE No.
DATE / /

Lab-3 DAA Assignment

Algorithm: for Creating a file C for Creating Data Entries of 2000 people.) CSV file

// Aim : Generating & adding random name & Salary.

```
String RandomGenratorName() {
    Vector<String> a = { // first Name }
    Vector<String> b = { // Surname }
}
String a = a[rand % a.Size()]
        // Random first Name is being selected.
Stringb = b[rand % b.Size()]
    // Random Surname is Chosen

int generateSalaryRandom() {
    // Random Salary Genrate.
    }
```

// Input : it is given in a format of void function
// output : it returns first & last name from the function we Created.

(2) Calculating Salary : Algorithm for Calculating Salary (HR, Bonus, etc)

// Aim : To Calculate taxes, bonus and HR and to Store it in a CSV file.
// input : Input file, Vector etc.

```
if ( !inputfile . is_open ) {
  //error }
outputfile = << "Name, Salary, Tax, HR, Bonus";

getline (input file, line);

While (get-line (inputfile, line)) {
   StringStream ss (line);
   get line (ss, Name, ',' ),
   getline (// Similar for Salary as name)

   tax = 0.30 * Salary
   HR = 0.12 * Salary
   Bonus 0.15 * Salary;

Close input file.
```

// function to find Maximum number Algorithm

~~int~~ f

```
function findMax(a[], lo, hi):
    if lo > hi:
        return INT_MIN
    if lo == hi:
        return a[lo]

    mid = (lo + hi) / 2

leftMax = findMax(a, lo, mid)
right = find_Max(a, mid+1, hi)

return max(left max, right)
```

Algorithm of Minimum number

```
function findMin(a[], lo, hi):
    if lo > hi:
        return INT_MIN
    if lo == hi:
        return a[lo]
    mid = (lo + hi) / 2

leftMim = find_Min(a, lo, mid)
right_Min = findMin(a, mid+1, hi)

return min(leftMin, rightMin).
```

PAGE No.
DATE / /

Time Complexity

By Substitution :-

$T(M) = 2 * \lceil (M/2) \rceil + M(M)$

Where: $T(K)$ is Time taken to Sort K ~~input~~ element.

// $M(K)$: time taken to merge K elements.

$$2 * T(M/2) + Const. * M \quad -(1)$$

$$T(M/2) = 2 * T(M/4) + Const\ M/2 \quad -(2)$$

Putting eq^n (2) in (1)

$$T(M) = 2^2 T(M/2^2) + Cost \cdot \left(M + \frac{M}{2}\right)$$

$$T(M) = 2^2 T(M/2^2) + Const \cdot M\left(1 + \frac{1}{2}\right)$$

$$T(M) = 2^K T\left(M/2^K\right) + Const\ M\left(1 + \frac{1}{2^1} + \frac{1}{2^2} + \cdots \frac{1}{2^{K-1}}\right)$$

for $2^K = M$

$K = \log M$

$$T(M) = M\,(T(1)) + M \log_2 M\ Cost.$$

$$T(M) = M + M \log M \cdot Const$$

$$\approx M \log M \cdot Const \approx M \log M.$$

PAGE No
DATE / /

Time Complexity is $n \log(n)$. for Merge Sort Algo.