****

**Southeast University**

**Department of Computer Science and Engineering (CSE)**

**School of Sciences and Engineering**

**Semester: (Spring, Year: 2025)**

**LAB REPORT NO**: **01**

**Course Title**: Algorithm Lab

**Course Code:** CSE266.14

**Batch**: 65

**Lab Experiment Name:** **Developing a system record for Employees.**

**Student Details**

|  |  |  |
| --- | --- | --- |
| **Name** | | **ID** |
| **1.** | Md. Mahmud Hossain | 2023200000799 |

**Submission Date : 26-04-25**

**Course Teacher’s Name : Maisha Muntaha**

|  |
| --- |
| **Lab Report Status**  **Marks: ………………………………… Signature:.....................**  **Comments:.............................................. Date:..............................** |

**1) TITLE OF THE LAB REPORT EXPERIMENT:**

Developing a system record for Employees.

**2) OBJECTIVES:**

* To implement and understand the working of various sorting algorithms (Insertion Sort, Selection Sort, Quick Sort, and Merge Sort).
* To handle tie-breakers by sorting based on Employee ID when salaries are equal.
* To modify the Quick Sort algorithm to use the first element as pivot.
* To compare and display sorted employee records in a structured format.

**3) PROCEDURE:**

1. Define a structure Employee containing:

* id (Employee ID)
* salary (Employee Salary)

1. Write a comparison function compare() to:

* Sort employees based on salary (descending).
* If salaries are equal, sort based on ID (ascending).

1. Implement the following sorting algorithms:

* Insertion Sort (modified for descending order using compare()).
* Selection Sort (modified for descending order using compare()).
* Quick Sort (modified to pick the first element as pivot).
* Merge Sort (modified for descending order using compare()).

1. Take input:

* Number of employees.
* Employee ID and salary for each employee.

1. Display sorting menu for user to choose a sorting method.
2. Sort the employee records based on user's choice.
3. Display the sorted employee records in the required format.

**4) IMPLEMENTATION:**

**Question:** Solve the following problem and submit as a lab report.  
1)You are developing a system record for employees. Where each record contain two tuple as (employee\_id, salary)

* Sort the employee according to the salary descending order ( both Merge and quick sort)
* If two employe record has the same salary , sort then by employee\_id ascending order
* Sample Input: (e1,50000),(e2, 4000), (e4,20000)
* Sample Output: (e1,50000),(e4, 20000), (e4, 4000).

**Code:**

**= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =**

**#include<iostream>**

**#include<string>**

**#define MAX 100**

**using namespace std;**

**struct Employee {**

**string id;**

**int salary;**

**};**

**bool compare(const Employee &a, const Employee &b) {**

**if (a.salary != b.salary)**

**return a.salary > b.salary;**

**else**

**return a.id < b.id;**

**}**

**void Merge(Employee arr[], int p, int q, int r) {**

**int n1 = q - p + 1;**

**int n2 = r - q;**

**Employee L[MAX], R[MAX];**

**for (int i = 0; i < n1; i++)**

**L[i] = arr[p + i];**

**for (int j = 0; j < n2; j++)**

**R[j] = arr[q + 1 + j];**

**int i = 0, j = 0, k = p;**

**while (i < n1 && j < n2) {**

**if (compare(L[i], R[j])) {**

**arr[k++] = L[i++];**

**} else {**

**arr[k++] = R[j++];**

**}**

**}**

**while (i < n1)**

**arr[k++] = L[i++];**

**while (j < n2)**

**arr[k++] = R[j++];**

**}**

**void mergeSort(Employee arr[], int p, int r) {**

**if (p < r) {**

**int q = p + (r - p) / 2;**

**mergeSort(arr, p, q);**

**mergeSort(arr, q + 1, r);**

**Merge(arr, p, q, r);**

**}**

**}**

**int Partition(Employee arr[], int low, int high) {**

**Employee pivot = arr[low];**

**int i = low + 1;**

**int j = high;**

**while (true) {**

**while (i <= high && compare(arr[i], pivot))**

**i++;**

**while (j >= low + 1 && compare(pivot, arr[j]))**

**j--;**

**if (i > j)**

**break;**

**swap(arr[i], arr[j]);**

**}**

**swap(arr[low], arr[j]);**

**return j;**

**}**

**void quickSort(Employee arr[], int low, int high) {**

**if (low < high) {**

**int pivotIndex = Partition(arr, low, high);**

**quickSort(arr, low, pivotIndex - 1);**

**quickSort(arr, pivotIndex + 1, high);**

**}**

**}**

**void insertionSort(Employee arr[], int n) {**

**for (int i = 1; i < n; i++) {**

**Employee key = arr[i];**

**int j = i - 1;**

**while (j >= 0 && !compare(arr[j], key)) {**

**arr[j + 1] = arr[j];**

**j--;**

**}**

**arr[j + 1] = key;**

**}**

**}**

**void selectionSort(Employee arr[], int n) {**

**for (int i = 0; i < n - 1; i++) {**

**int id = i;**

**for (int j = i + 1; j < n; j++) {**

**if (!compare(arr[id], arr[j])) {**

**id = j;**

**}**

**}**

**if (id != i) {**

**swap(arr[i], arr[id]);**

**}**

**}**

**}**

**void displayEmployees(Employee arr[], int n) {**

**cout << "\nSorted Employee:\n";**

**for (int i = 0; i < n; ) {**

**cout << "(" << arr[i].id << arr[i].salary << ")";**

**i++;**

**if(i==n)**

**cout<<".";**

**else**

**cout<<",";**

**}**

**cout << endl;**

**}**

**int main() {**

**int n, choice;**

**Employee employees[MAX];**

**cout << "Number of employees: ";**

**cin >> n;**

**cout << "Enter Employee ID and Salary:\n";**

**for (int i = 0; i < n; i++) {**

**cin >> employees[i].id >> employees[i].salary;**

**}**

**cout << "\nSorting Options:\n";**

**cout << "1. Insertion Sort\n";**

**cout << "2. Selection Sort\n";**

**cout << "3. Quick Sort\n";**

**cout << "4. Merge Sort\n";**

**cout << "Press a option: ";**

**cin >> choice;**

**switch (choice) {**

**case 1:**

**insertionSort(employees, n);**

**displayEmployees(employees, n);**

**break;**

**case 2:**

**selectionSort(employees, n);**

**displayEmployees(employees, n);**

**break;**

**case 3:**

**quickSort(employees, 0, n - 1);**

**displayEmployees(employees, n);**

**break;**

**case 4:**

**mergeSort(employees, 0, n - 1);**

**displayEmployees(employees, n);**

**break;**

**default:**

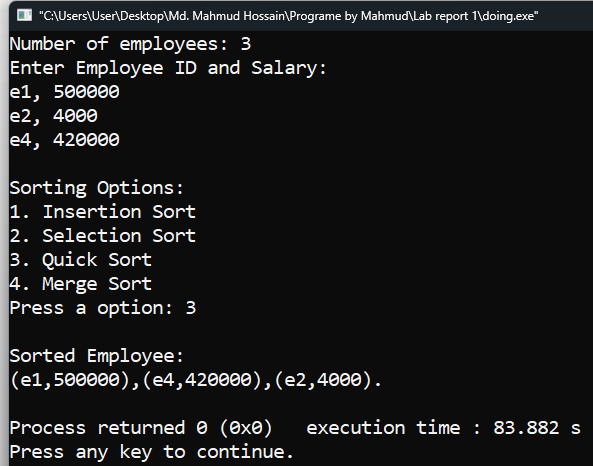
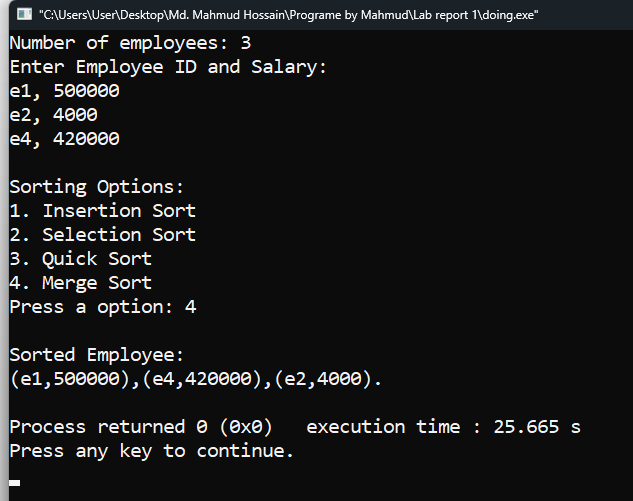
**cout << "Invalid choice\n";**

**break;**

**}**

**return 0; }**

**4) OUTPUT:**



**5) DISCUSSION:**

Finally, I got my result. In this report, I created a C++ program that sorts employee records by using different sorting algorithms. I used structures to store employee data and applied Insertion Sort, Selection Sort, Quick Sort, and Merge Sort based on the user’s choice. The main challenge was adjusting the Quick Sort to use the first element as the pivot instead of the last one, but after modifying the partition logic, it worked properly. Another tricky part was comparing salary first and then ID when salaries were the same. However, my main concern was to correctly sort the employee records in descending order and show them properly, and I did it successfully.