**Department of Computing**

**School of Electrical Engineering and Computer Science**

**CS250 – Data Structures and Algorithms**



**Lab 13: Open-ended Lab**

**Submission Details**

|  |  |
| --- | --- |
| Name | CMS ID |
| Muhammad Umer | 345834 |
| Group | GP – 1 |
| Lab Engineer | Anum Asif |
| Faculty Member | Bostan Khan |
| Class | BEE12 |
| Date | 06/05/2024 |
| Time | 10:00 am – 12:50 pm |

**Table of Contents**

[2 Open-ended Lab 3](#_Toc165908401)

[2.1 Introduction 3](#_Toc165908402)

[2.2 Objectives 3](#_Toc165908403)

[2.3 Tools/Software Requirement 3](#_Toc165908404)

[2.4 Deliverables 3](#_Toc165908405)

[2.5 Description 3](#_Toc165908406)

[3 Lab Tasks 4](#_Toc165908407)

# Open-ended Lab

## Introduction

This is an open ended lab where you have to choose your own unique dataset and implement your own methodology based on your understanding of the problem.

## Objectives

Objective of this open ended lab is to implement a problem using the methodology of your choice.

## Tools/Software Requirement

* Visual Studio C++

## Deliverables

Compile a single word document by filling in the solution parts and submit this file on LMS. The name of word document should follow this format. i.e., YourFullName(reg)\_Lab#. You must show the implementation of the tasks in a complete manner to get your work graded.

## Description

The basic idea of the task is that you have to take the grades of the courses you took in previous semester. The grades should be taken for a number of students, means for every individual course e.g., OOP, the list of grades will be created for a total of e.g., 40 students. Then you have to sort each individual list of grades, corresponding to each course. Once the individual lists are sorted, you have to merge all the lists.

# Lab Tasks

Implement the problem described above. Make sure to use your own and unique dataset and sorting algorithm. An appropriate choice of **data structure to store grades** is required. Also, **efficient and appropriate sorting** algorithm is to be used for sorting purpose.

*Note: Mention the reasoning for the choice of data structure and sorting algorithms used.*

Code

#include <fstream>

#include <iostream>

#include <sstream>

#include <string>

using *namespace* std;

*const* *int* MAX\_ROWS = 4 \* 20; *// Maximum number of rows in the dataset*

*// Heapify function for heap sort*

*void* heapify(string *student\_data*[MAX\_ROWS][3], *int* *n*, *int* *i*) {

*int* largest = *i*;

*int* l = 2 \* *i* + 1;

*int* r = 2 \* *i* + 2;

    if (l < *n* && *student\_data*[l][2] > *student\_data*[largest][2])

        largest = l;

    if (r < *n* && *student\_data*[r][2] > *student\_data*[largest][2])

        largest = r;

    if (largest != *i*) {

        swap(*student\_data*[*i*], *student\_data*[largest]);

        heapify(*student\_data*, *n*, largest);

    }

}

*// Heap sort for sorting grades*

*void* heap\_sort(string *student\_data*[MAX\_ROWS][3], *int* *n*) {

    for (*int* i = *n* / 2 - 1; i >= 0; i--)

        heapify(*student\_data*, *n*, i);

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

        swap(*student\_data*[0], *student\_data*[i]);

        heapify(*student\_data*, i, 0);

    }

}

*// Read, sort, and merge the dataset*

*void* read\_sort\_merge(*const* string*&* *filename*, string *merged*[MAX\_ROWS][3], *int&* *merged\_rows*) {

    ifstream file(*filename*);

    if (!file.is\_open()) {

        cout << "Unable to open file";

        return;

    }

    string student\_data[MAX\_ROWS][3];

*int* rows = 0;

    string line;

    while (getline(file, line) && rows < MAX\_ROWS) {

        istringstream ss(line); *// Convert the line into string stream*

*// Read data until comma, store in student\_data[...]*

        getline(ss, student\_data[rows][0], ',');

        getline(ss, student\_data[rows][1], ',');

        getline(ss, student\_data[rows][2], ',');

        ++rows;

    }

    file.close();

*// Sort student data*

*int* start = 0;

    while (start < rows) {

*int* end = start + 4;

        heap\_sort(student\_data + start, 4);

        for (*int* i = start; i < end; ++i) {

            for (*int* j = 0; j < 3; ++j) {

*merged*[*merged\_rows*][j] = student\_data[i][j];

            }

            ++*merged\_rows*;

        }

        start = end;

    }

}

*// Write the dataset as table*

*void* write\_as\_table(*const* string*&* *filename*, *const* string *data*[MAX\_ROWS][3], *int* *rows*) {

    ofstream file(*filename*);

    if (!file.is\_open()) {

        cout << "Unable to open file";

        return;

    }

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

        file << *data*[i][0] << "," << *data*[i][1] << "," << *data*[i][2] << "\n";

    }

    file.close();

}

*int* main() {

    string merged[MAX\_ROWS][3]; *// Assuming maximum rows for simplicity*

*int* merged\_rows = 0;

    read\_sort\_merge("dataset.txt", merged, merged\_rows);

    write\_as\_table("merged.txt", merged, merged\_rows);

    return 0;

}

Output

**Original (dataset.txt)**

Ayesha Khan,CV,C

Ayesha Khan,EPM,A

Ayesha Khan,ML,B

Ayesha Khan,MCS,A

Muhammad Rizwan,CV,C

Muhammad Rizwan,EPM,A

Muhammad Rizwan,ML,A

Muhammad Rizwan,MCS,A

Sameera Bashir,CV,A

Sameera Bashir,EPM,A

...

...

Nadia Imran,ML,D

Nadia Imran,MCS,A

Junaid Muhammad,CV,B

Junaid Muhammad,EPM,A

Junaid Muhammad,ML,A

Junaid Muhammad,MCS,D

**Sorted (merged.txt)**

Ayesha Khan,EPM,A

Ayesha Khan,MCS,A

Ayesha Khan,ML,B

Ayesha Khan,CV,C

Muhammad Rizwan,EPM,A

Muhammad Rizwan,ML,A

Muhammad Rizwan,MCS,A

Muhammad Rizwan,CV,C

Sameera Bashir,EPM,A

Sameera Bashir,CV,A

...

...

Nadia Imran,ML,D

Nadia Imran,CV,D

Junaid Muhammad,EPM,A

Junaid Muhammad,ML,A

Junaid Muhammad,CV,B

Junaid Muhammad,MCS,D

Reason/Justification of Implementation Choices

Heap sort is the optimal choice for this task due to its efficient average-case time complexity, which ensures reliable performance even when dealing with varying grades across multiple students and courses. Unlike other sorting algorithms like insertion, bubble, and selection sorts, heap sort handles potential variations in grades effectively, thanks to its balanced time complexity of . Moreover, using heap sort over merge sort is preferred due to lower memory footprint, i.e., heap sort has a space complexity of compared to of merge sort.

The decision to use static arrays is based on the known length of the dataset beforehand, with 20 students each having 4 courses, totaling 80 rows. Static arrays offer a straightforward and efficient solution without the need for dynamic memory management, simplifying implementation and ensuring optimal memory allocation. By combining heap sort with static arrays, we achieve a practical and effective approach for sorting student grades in this context.