**Operating System – Lab 02**

**Name:** Hafsa Salman

**Roll no.** 22K-5161

**In-Lab Questions**

**Task no. 01**

Code:

#include<stdio.h>

#include<stdlib.h>

int main(int argc, char \*argv[])

{

FILE \*file\_ptr;

char ch;

if (argc != 2)

{

printf("Usage: %s <filename>\n", argv[0]);

return 1;

}

file\_ptr = fopen(argv[1], "r");

if (file\_ptr == NULL)

{

perror("Error");

return 1;

}

printf("Contents of %s: \n", argv[1]);

while ((ch = fgetc(file\_ptr)) != EOF)

{

putchar(ch);

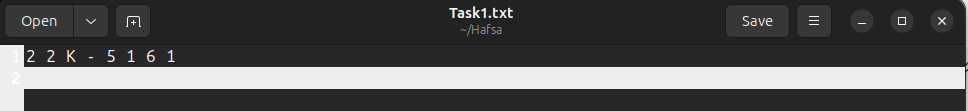
}

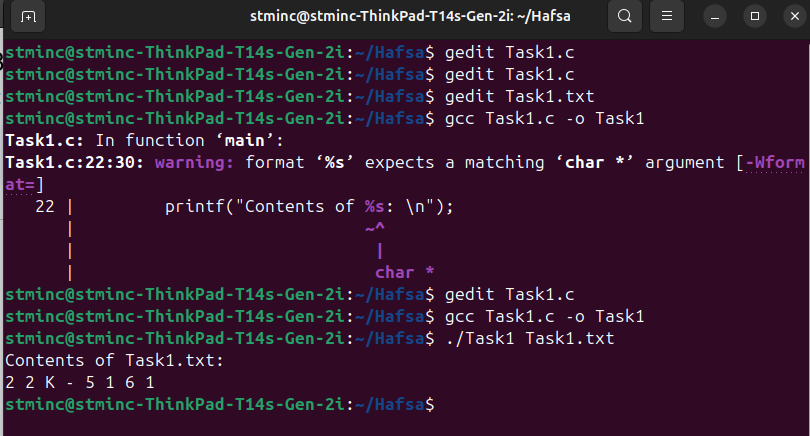
fclose(file\_ptr);

return 0;

}

Content of Task1.txt:



Output:

**Task no. 02**

Code:

#include<stdio.h>

#include<stdlib.h>

int main (int argc, char \*argv[])

{

if (argc < 2)

{

printf("Usage: %s <integer1><integer2> ... <integerN>\n", argv[0]);

return 1;

}

int integers[argc - 1];

for (int i=1; i<argc; i++)

{

integers[i-1] = atoi(argv[i]);

}

for (int i=0; i<argc-2; i++)

{

for (int j=0; j<argc-i-2; j++)

{

if (integers[j] > integers[j+1])

{

int temp = integers[j];

integers[j] = integers[j+1];

integers[j+1] = temp;

}

}

}

printf("Sorted Integer: ");

for (int i=0; i<argc-1; i++)

{

printf("%d ", integers[i]);

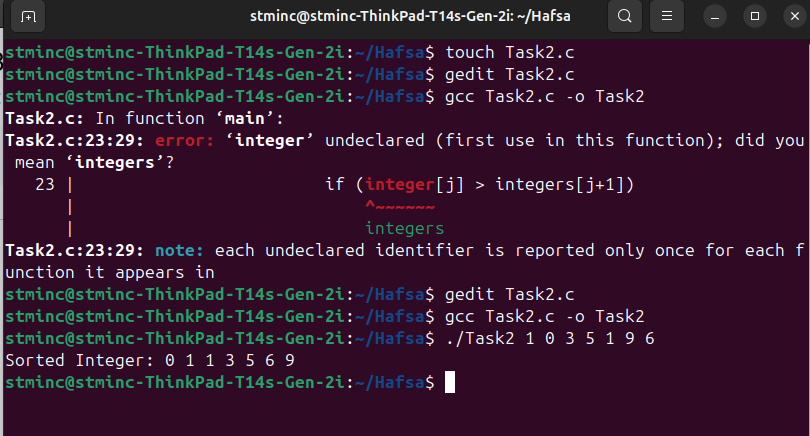
}

printf("\n");

return 0;

}

Output:



**Task no. 03**

Code:

//Student.cpp

#include "Student.h"

#include "Course.h"

#include <iostream>

Student::Student(std::string name) : name(name) {}

void Student::enroll(Course\* course)

{

coursesEnrolled.push\_back(course);

}

void Student::displayCourses()

{

std::cout << "Courses enrolled by " << name <<":\n";

for (Course\* course: coursesEnrolled)

{

std::cout << course->getName() << "\n";

}

}

std::string Student::getName() const

{

return name;

}

//Teacher.cpp

#include "Teacher.h"

#include"Course.h"

#include <iostream>

Teacher::Teacher(std::string name) : name(name) {}

void Teacher::teach(Course\* course)

{

coursesTeaching.push\_back(course);

}

void Teacher::displayCourses()

{

std::cout << "Courses taught by " << name << ":\n";

for (Course\* course : coursesTeaching)

{

std::cout << course->getName() <<"\n";

}

}

std::string Teacher::getName() const

{

return name;

}

//Course.cpp

#include "Course.h"

#include "Student.h"

#include "Teacher.h"

#include <iostream>

Course::Course(std::string name) : name(name) {}

std::string Course::getName() const

{

return name;

}

void Course::addStudent(Student\* student)

{

students.push\_back(student);

}

void Course::addTeacher(Teacher\* teacher)

{

teachers.push\_back(teacher);

}

void Course::displayStudents()

{

std::cout << "Students studying " << name << ":\n";

for (Student\* student : students)

{

std::cout << student->getName() << "\n";

}

}

void Course::displayTeachers()

{

std::cout << "Teachers teaching " << name << ":\n";

for (Teacher\* teacher : teachers)

{

std::cout << teacher->getName() << "\n";

}

}

//Student.h

#ifndef STUDENT\_H

#define STUDENT\_H

#include <string>

#include <vector>

class Course;

class Student

{

private:

std::string name;

std::vector<Course\*> coursesEnrolled;

public:

Student(std::string name);

void enroll(Course\* course);

void displayCourses();

std::string getName() const;

};

#endif

//Teacher.h

#ifndef TEACHER\_H

#define TEACHER\_H

#include <string>

#include <vector>

class Course;

class Teacher

{

private:

std::string name;

std::vector<Course\*> coursesTeaching;

public:

Teacher(std::string name);

void teach(Course\* course);

void displayCourses();

std::string getName() const;

};

#endif

//Course.h

#ifndef COURSE\_H

#define COURSE\_H

#include <string>

#include <vector>

class Student;

class Teacher;

class Course

{

private:

std::string name;

std::vector<Student\*> students;

std::vector<Teacher\*> teachers;

public:

Course(std::string name);

std::string getName() const;

void addStudent(Student\* student);

void addTeacher(Teacher\* teacher);

void displayStudents();

void displayTeachers();

};

#endif

//Main.cpp

#include <iostream>

#include "Student.h"

#include "Teacher.h"

#include "Course.h"

int main()

{

Student s1("Hafsa");

Teacher t1 ("Mr. Sameer Faisal");

Course c1 ("OS Lab");

s1.enroll(&c1);

t1.teach(&c1);

s1.displayCourses();

t1.displayCourses();

c1.displayStudents();

c1.displayTeachers();

return 0;

}

//Makefile

CXX = g++

CXXFLAGS = -std=c++11

all: main

main: main.o Student.o Teacher.o Course.o

$(CXX) $(CXXFLAGS) main.o Student.o Teacher.o Course.o -o main

main.o: main.cpp Student.h Teacher.h Course.h

$(CXX) $(CXXFLAGS) -c main.cpp

Student.o: Student.cpp Student.h Course.h

$(CXX) $(CXXFLAGS) -c Student.cpp

Teacher.o: Teacher.cpp Teacher.h Course.h

$(CXX) $(CXXFLAGS) -c Teacher.cpp

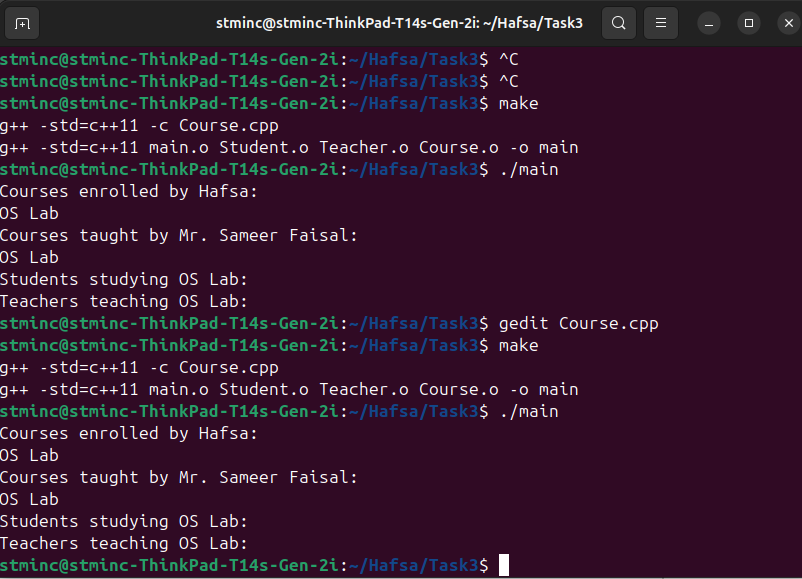
Course.o: Course.cpp Course.h Student.h Teacher.h

$(CXX) $(CXXFLAGS) -c Course.cpp

clean:

rm -rf \*.o main

Output:



**Post Lab Questions**

**Task no. 01**

#include <iostream>

#include <fstream>

#include <algorithm>

using namespace std;

int main()

{

int arr[10];

cout << "Enter 10 integers: ";

for (int i=0; i<10; ++i)

{

cin >> arr[i];

}

sort(arr, arr+10);

ofstream outFile("Task1.txt");

if (outFile.is\_open())

{

for (int i=0; i<10; ++i)

{

outFile << arr[i] << " ";

}

outFile.close();

cout << "Array is sorted and saved in the file.";

}

else

{

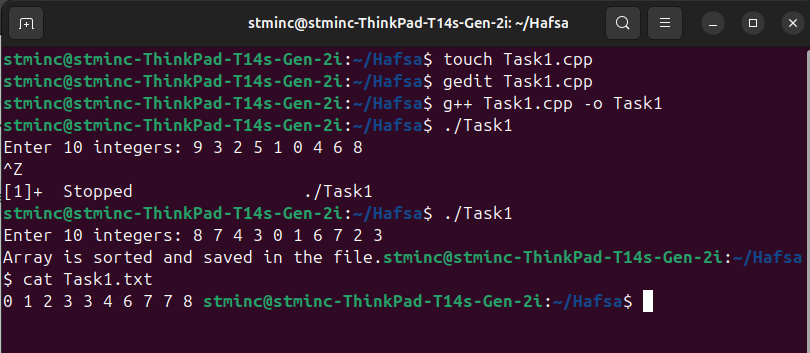
cout << "Unable to open file.";

}

return 0;

}

Output



**Task no. 02**

Code:

#include <iostream>

#include <fstream>

#include <vector>

#include <algorithm>

using namespace std;

struct Node

{

int data;

Node\* prev;

Node\* next;

};

void insert(Node\*\* head, int data)

{

Node\* newNode = new Node();

newNode->data = data;

newNode->next = nullptr;

if (\*head == nullptr)

{

newNode->prev = nullptr;

\*head = newNode;

return;

}

Node\* temp = \*head;

while (temp->next != nullptr)

{

temp = temp->next;

}

temp->next = newNode;

newNode->prev = temp;

}

void sortList(Node\*\* head)

{

vector<int> dataList;

Node\* temp = \*head;

while (temp != nullptr)

{

dataList.push\_back(temp->data);

temp = temp->next;

}

sort(dataList.begin(), dataList.end());

temp = \*head;

int i = 0;

while (temp != nullptr)

{

temp->data = dataList[i++];

temp = temp->next;

}

}

void writeToFile(Node\* head, const string& filename)

{

ofstream outFile(filename);

if (!outFile)

{

cerr << "Failed to open file: " << filename << endl;

return;

}

Node\* temp = head;

while (temp != nullptr)

{

outFile << temp->data << " ";

temp = temp->next;

}

outFile << endl;

outFile.close();

}

int main()

{

Node\* head = nullptr;

int num;

cout << "Enter integers for the list (-1 to end): ";

while (cin >> num && num != -1)

{

insert(&head, num);

}

sortList(&head);

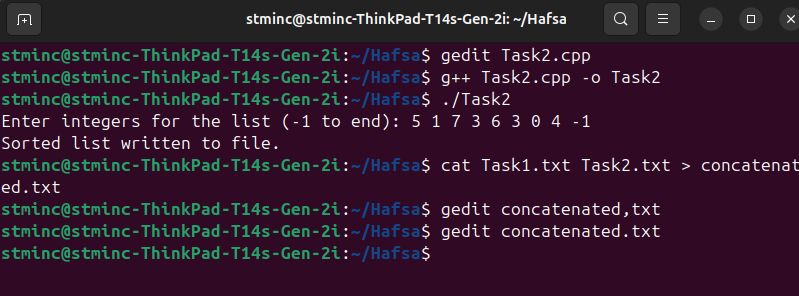
writeToFile(head, "Task2.txt");

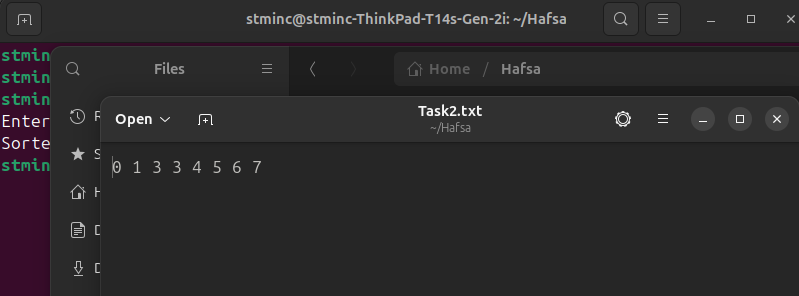
cout << "Sorted list written to file." << endl;

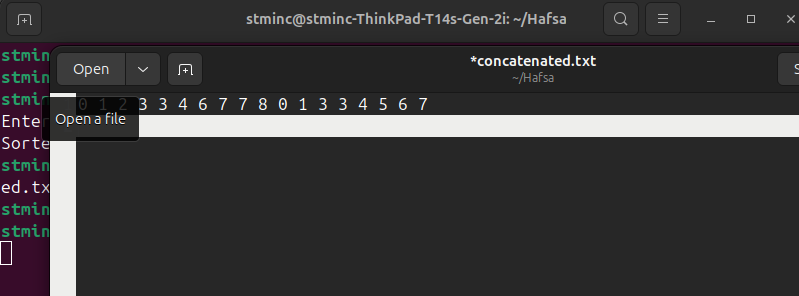
return 0;

}

Output:







**Task no. 03**

**Code:**

#include <iostream>

#include <fstream>

#include <vector>

#include <numeric>

#include <algorithm>

using namespace std;

int findMissingElement(const vector<int>& series)

{

int n = series.size() + 1;

int totalSum = (n \* (n+1))/2;

int seriesSum = accumulate(series.begin(), series.end(), 0);

return totalSum - seriesSum;

}

int main()

{

vector<int> series;

int num;

ifstream inFile("Task3.txt");

if (!inFile.is\_open())

{

cout << "Unable to open file.\n";

return 1;

}

while (inFile >> num)

{

series.push\_back(num);

}

inFile.close();

int missingElement = findMissingElement(series);

ofstream outFile("missing.txt");

if(!outFile.is\_open())

{

cout << "Unable to create file.\n";

return 1;

}

outFile << "The missing element is: " <<missingElement <<endl;

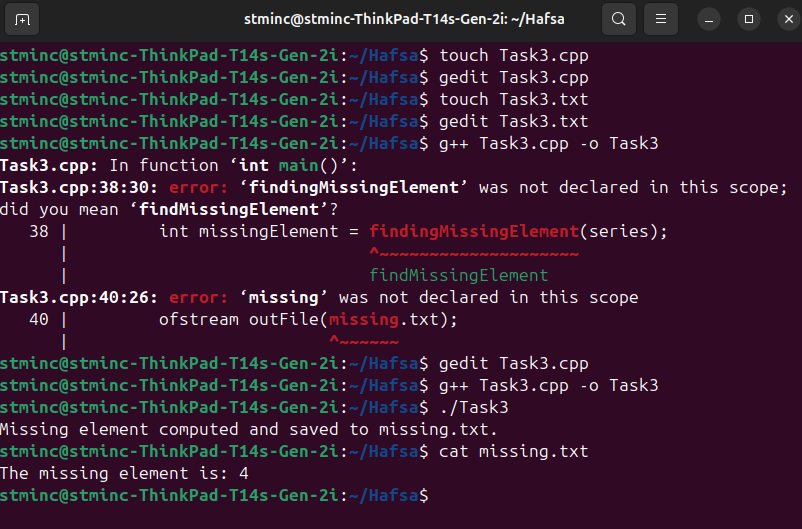
outFile.close();

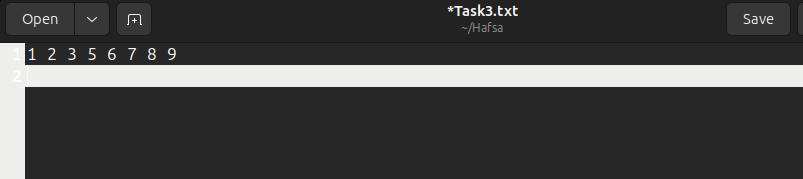
cout << "Missing element computed and saved to missing.txt.\n";

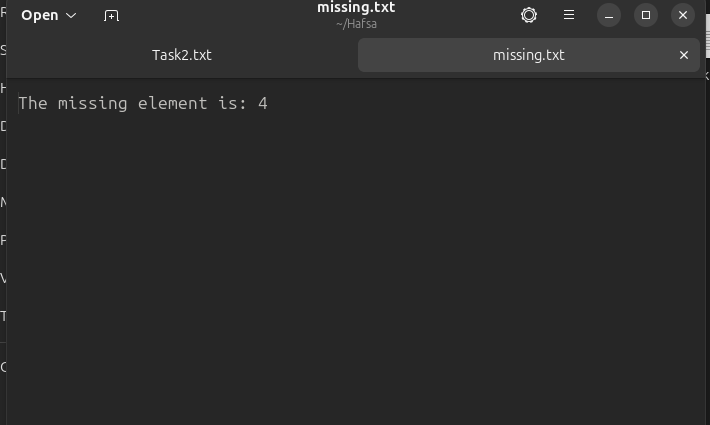
return 0;

}

Output:







**Task no. 04**

Code:

//Makefile

CC = g++

CFLAGS = -std=c++11

all: main

main: main.o LetterCount.o WordCount.o LineCount.o

$(CC) $(CFLAGS) -o main main.o LetterCount.o WordCount.o LineCount.o

main.o: main.cpp LetterCount.h WordCount.h LineCount.h

$(CC) $(CFLAGS) -c main.cpp

LetterCount.o: LetterCount.cpp LetterCount.h

$(CC) $(CFLAGS) -c LetterCount.cpp

WordCount.o: WordCount.cpp WordCount.h

$(CC) $(CFLAGS) -c WordCount.cpp

LineCount.o: LineCount.cpp LineCount.h

$(CC) $(CFLAGS) -c LineCount.cpp

clean:

rm -rf \*.o main

//LetterCount.h

#ifndef LETTERCOUNT\_H

#define LETTERCOUNT\_H

#include <fstream>

#include <iostream>

class LetterCount

{

public:

LetterCount(const std::string& filename);

int countLetters();

private:

std::string filename;

};

#endif

//LetterCount.cpp

#include "LetterCount.h"

LetterCount::LetterCount(const std:: string& filename) : filename(filename) {}

int LetterCount::countLetters()

{

std::ifstream file(filename);

if(!file.is\_open())

{

std::cerr << "Error opening file: " << filename << std::endl;

return -1;

}

int count = 0;

char ch;

while (file.get(ch))

{

if (isalpha(ch))

{

count++;

}

}

file.close();

return count;

}

//WordCount.h

#ifndef WORDCOUNT\_H

#define WORDCOUNT\_H

#include<fstream>

#include<iostream>

#include<string>

class WordCount

{

public:

WordCount(const std::string& filename);

int countWords();

private:

std::string filename;

};

#endif

//WordCount.cpp

#include "WordCount.h"

WordCount::WordCount(const std::string& filename) : filename(filename) {}

int WordCount::countWords()

{

std::ifstream file(filename);

if(!file.is\_open())

{

std::cerr << "Error opening file: " << filename << std::endl;

return -1;

}

int count = 0;

std::string word;

while (file >> word)

{

count++;

}

file.close();

return count;

}

//LineCount.h

#ifndef LINECOUNT\_H

#define LINECOUNT\_H

#include <fstream>

#include <iostream>

#include <string>

class LineCount

{

public:

LineCount(const std::string& filename);

int countLines();

private:

std::string filename;

};

#endif

//LineCount.cpp

#include "LineCount.h"

LineCount::LineCount(const std::string& filename) : filename(filename) {}

int LineCount::countLines()

{

std::ifstream file(filename);

if(!file.is\_open())

{

std::cerr << "Error opening file: " << filename << std::endl;

return -1;

}

int count = 0;

std::string line;

while (std::getline(file, line))

{

count++;

}

file.close();

return count;

}

//Main.cpp

#include <iostream>

#include "LetterCount.h"

#include "WordCount.h"

#include "LineCount.h"

int main()

{

LetterCount letterCounter("input.txt");

WordCount wordCounter("input.txt");

LineCount lineCounter("input.txt");

int letterCount = letterCounter.countLetters();

int wordCount = wordCounter.countWords();

int lineCount = lineCounter.countLines();

std::cout << "No. of letters: " << letterCount << std::endl;

std::cout << "No. of words: " << wordCount << std:: endl;

std::cout << "No. of lines: " << lineCount << std::endl;

return 0;

}

Output:

