# 

**Mazen Ahmed Darweish 2020/11224**

**Reem Raouf Metwally 2020/09601**

**Student Management System Report**

**Contents**

[Introduction 2](#_Toc160739576)

[Variables: 3](#_Toc160739577)

[Operation 1: **LOGIN** 4](#_Toc160739578)

[Operation 2: **ADD STUDENT** 5](#_Toc160739579)

[1. Student name 5](#_Toc160739580)

[2. Student ID 6](#_Toc160739581)

[3. Academic year 7](#_Toc160739582)

[4. GPA 7](#_Toc160739583)

[5. Check gender 7](#_Toc160739584)

[Operation 3: **REMOVE** 8](#_Toc160739585)

[1. Check length: 8](#_Toc160739586)

[2. Check validity: 9](#_Toc160739587)

[Operation 4: **Retrieve Student** 10](#_Toc160739588)

[1. Method 1: Search by name 10](#_Toc160739589)

[2. Method 2: Search by ID 11](#_Toc160739590)

[Operation 5: **Update** 12](#_Toc160739591)

[For updating GPA: 12](#_Toc160739592)

[For updating Academic Year: 13](#_Toc160739593)

[Operation 6: **Return** 14](#_Toc160739594)

[Problems faced: 15](#_Toc160739595)

[Conclusion: 15](#_Toc160739596)

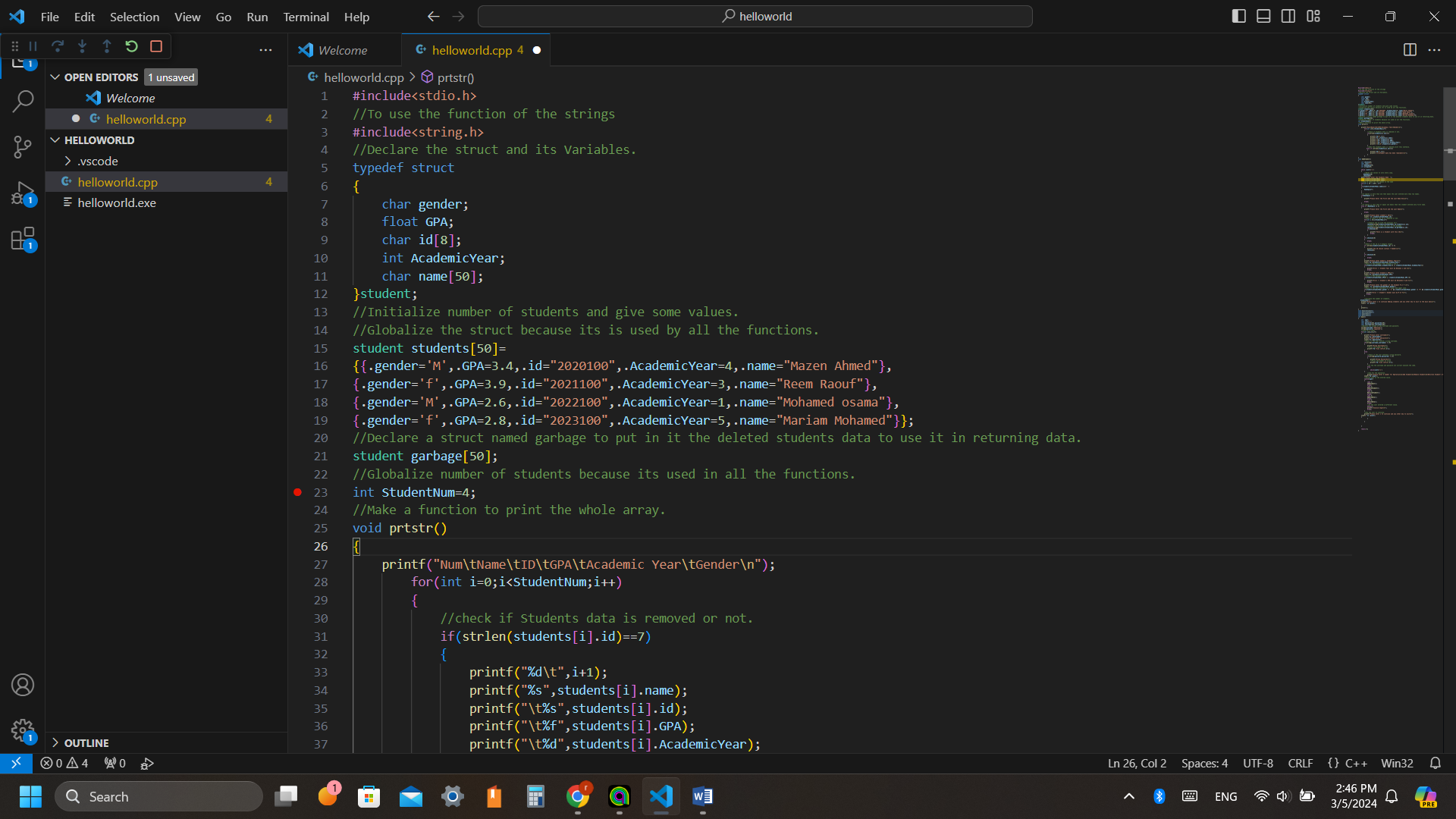
# Introduction

This report outlines the design and functionality of a student management system using the C programming language in Visual Studio Code, providing a brief overview of its 6 key operations:

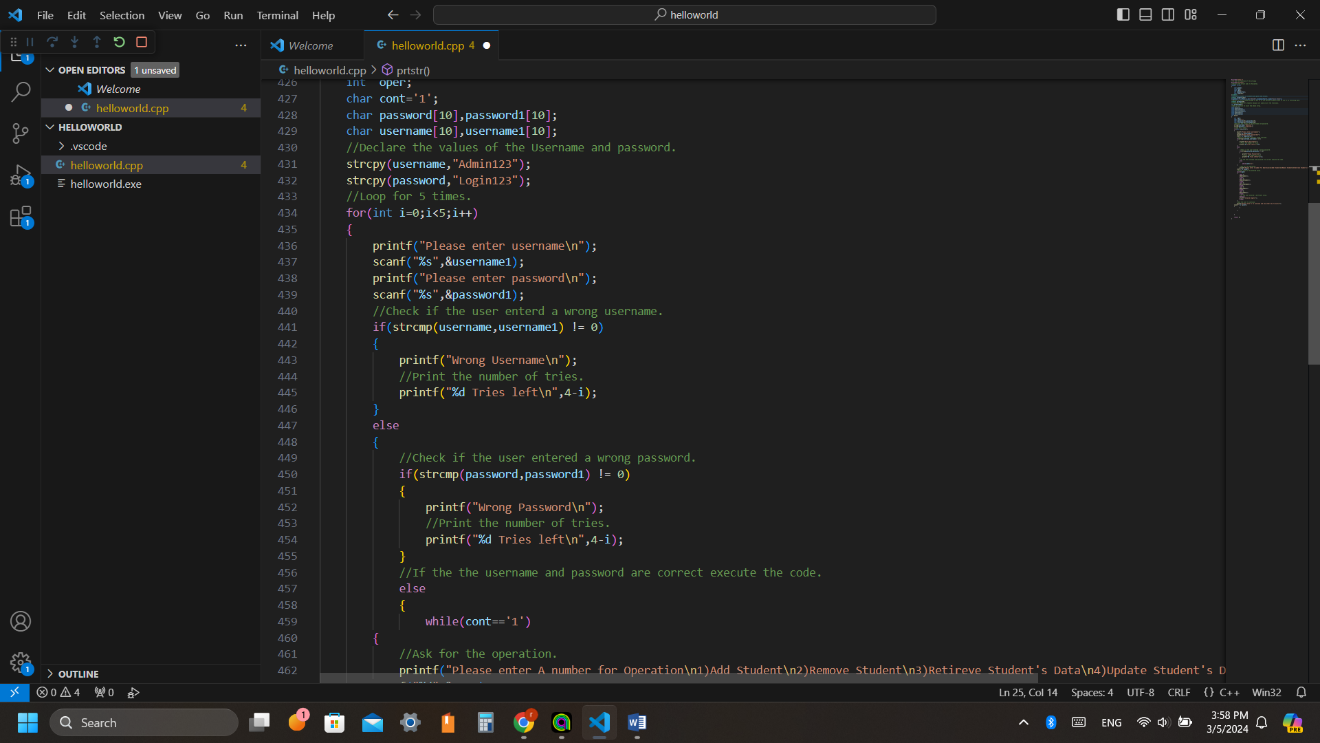
1. Login
2. Add Student
3. Remove Student
4. Retrieve Student's data
5. Update Student's data
6. Return Student's data

# Variables:

1. Declare structure named student, of 5 members of the structure: [Gender, GPA, Student ID, Academic Year, Name]
2. Initialize number of students to be 50, and set data of some students
3. Declare another structure named garbage of size 50; to store in it the deleted students’ data to restore it if needed.



# Operation 1: **LOGIN**

* The system requires form the user to enter Username and Password to login the system and start operating.
* The user has only 5 trials to enter wrong username or password, if the 5 attempts run out, the system will close.
* This is done by using the string comparing function (strcpy), which compares the user inputs with the declared username and password.

Username: Admin123

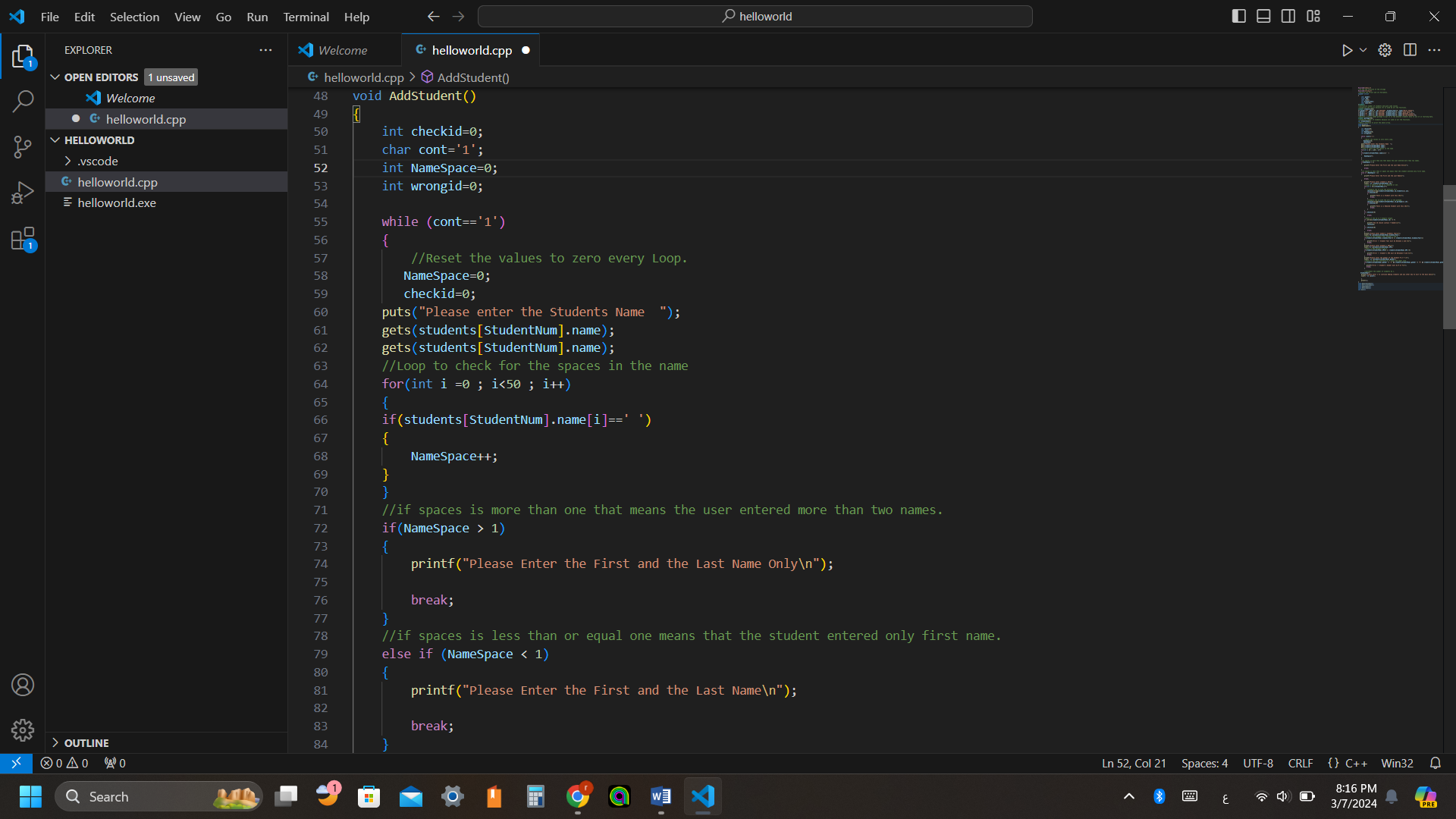
Password: Login123

# Operation 2: **ADD STUDENT**

Adding a student will take the previously mentioned 5 information, the system will do 5 error checks:

## Student name

* To check if the user entered his first and last names, not less or more.
* This done by counting spaces between the names; if spaces<1 this means he entered one name, if spaces>1 this means he entered more than two names.
* In both cases the system gives an error message.

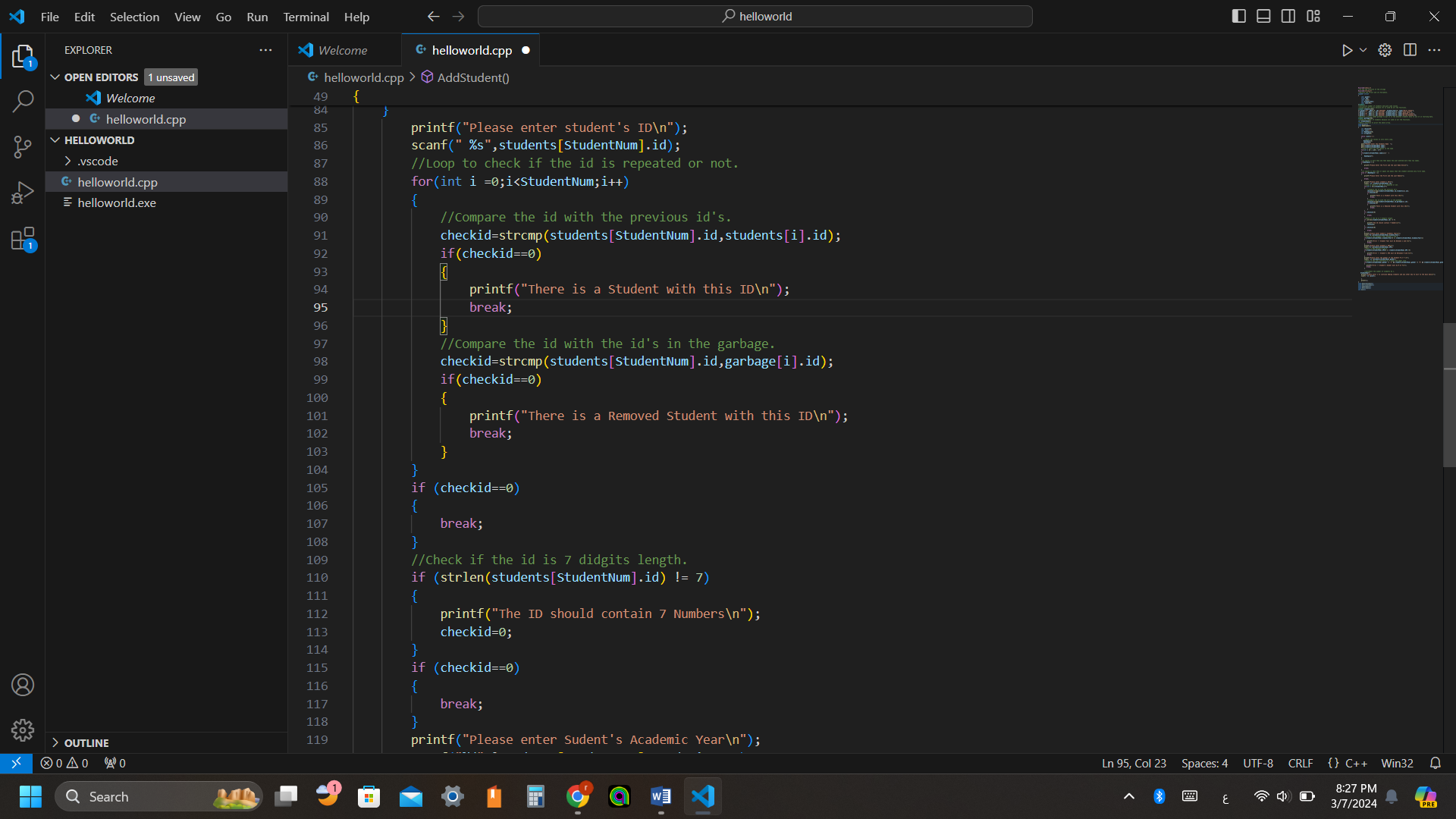


## Student ID

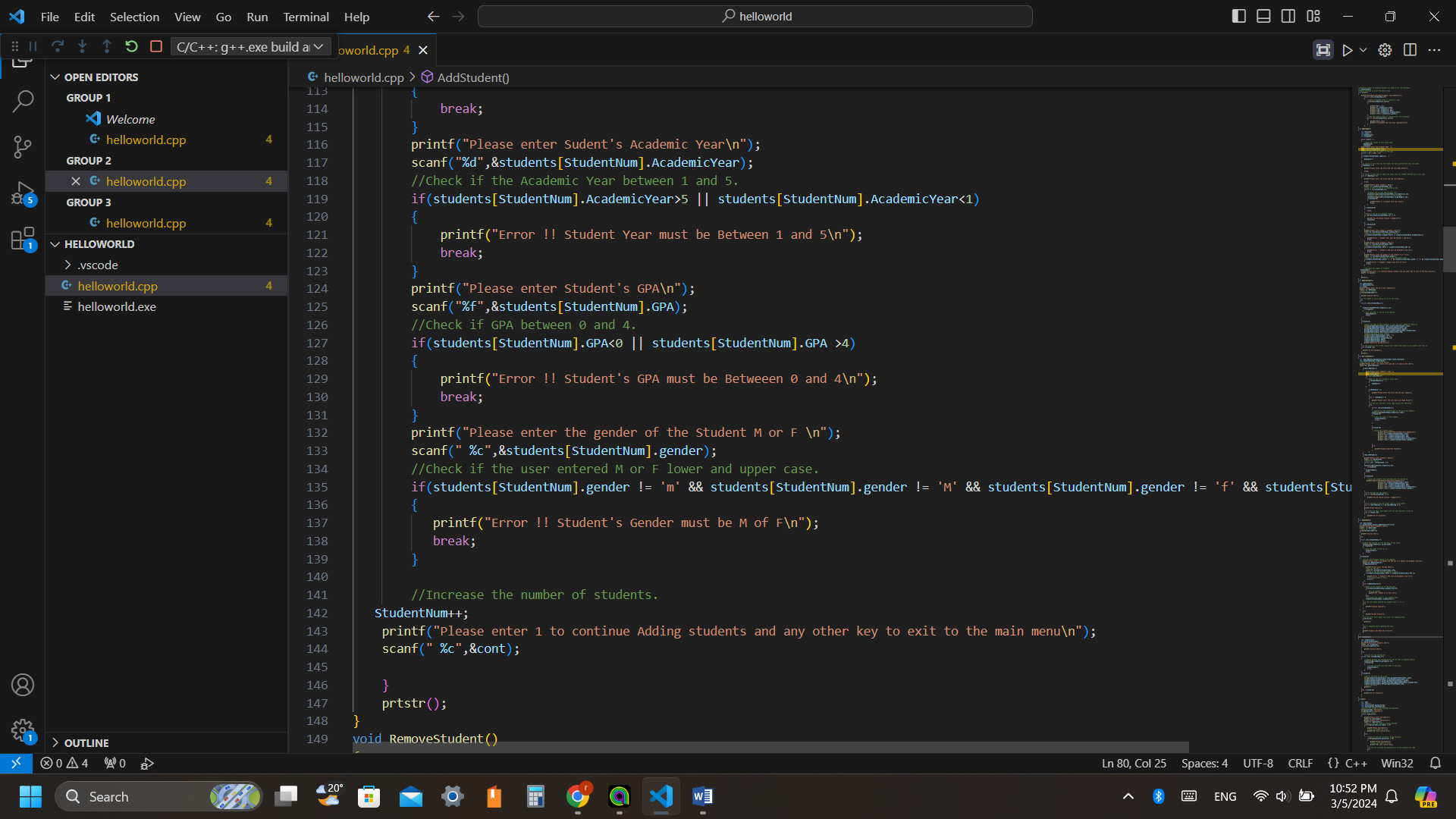
* Check if the ID is repeated or not, either in the active students’ data or the deleted data in the garbage.
* This done by comparing using (strcmp) function the given ID with all the ID’s in the students and garbage structures.

\*\*\*\*\*\*\*\*\*\*\*\*

* Or check the length of ID that must be 7 digits.
* This done by checking the ID length, if length not =7 will print error message.

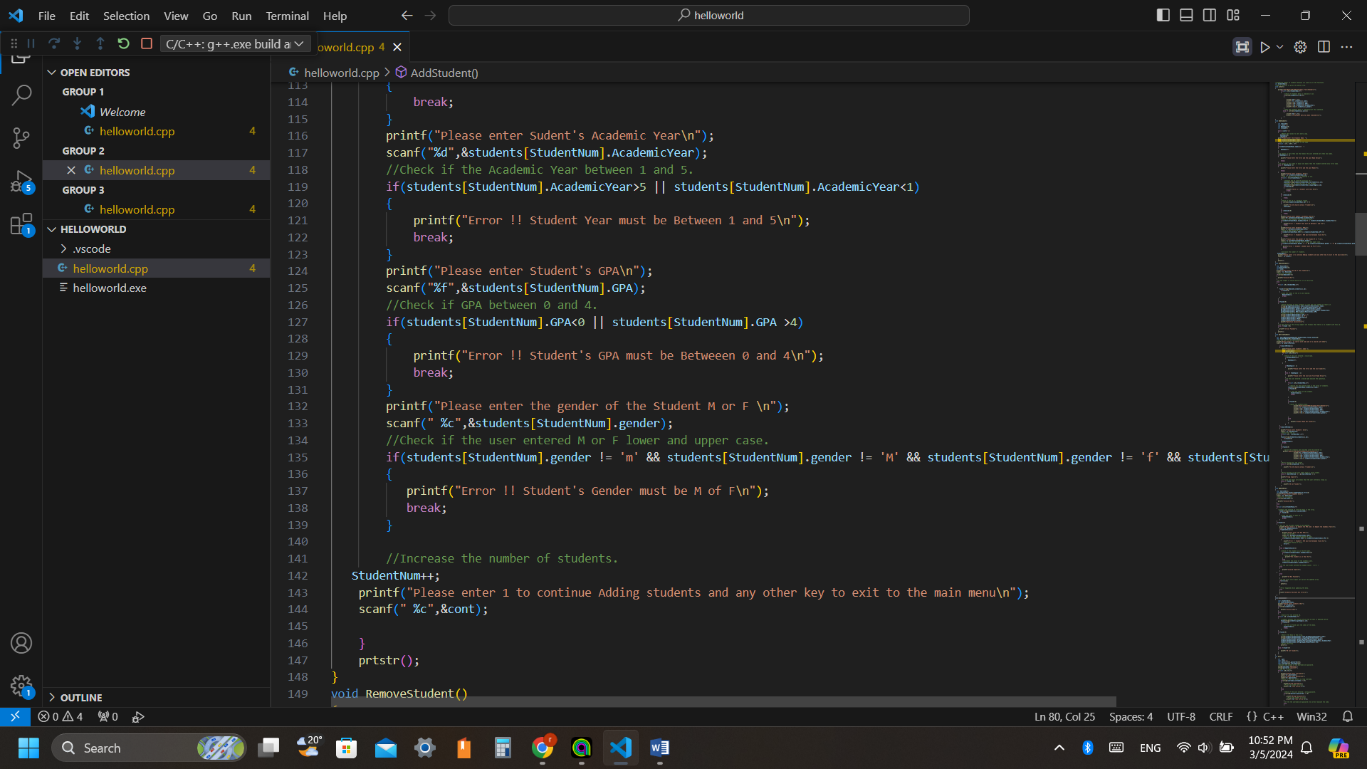


## Academic year

* Check if the academic year from 1 year to 5 years.
* If the user entered an input out of this interval an error message warns the user.

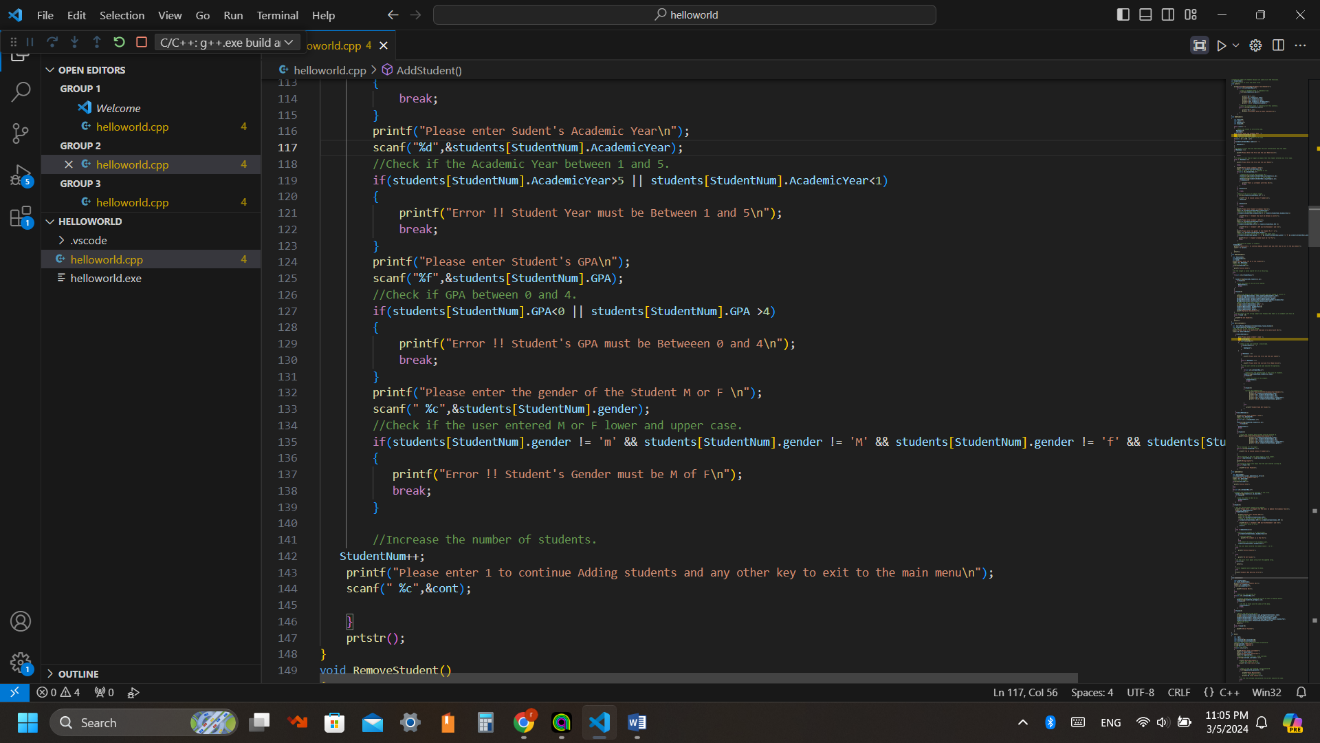
\*\*\*\*\*\*\*\*\*\*\*\*

## GPA

* Check if the entered student’s GPA from 0 to 4.
* If the user entered an input out of this interval an error message warns the user.

\*\*\*\*\*\*\*\*\*\*\*\*

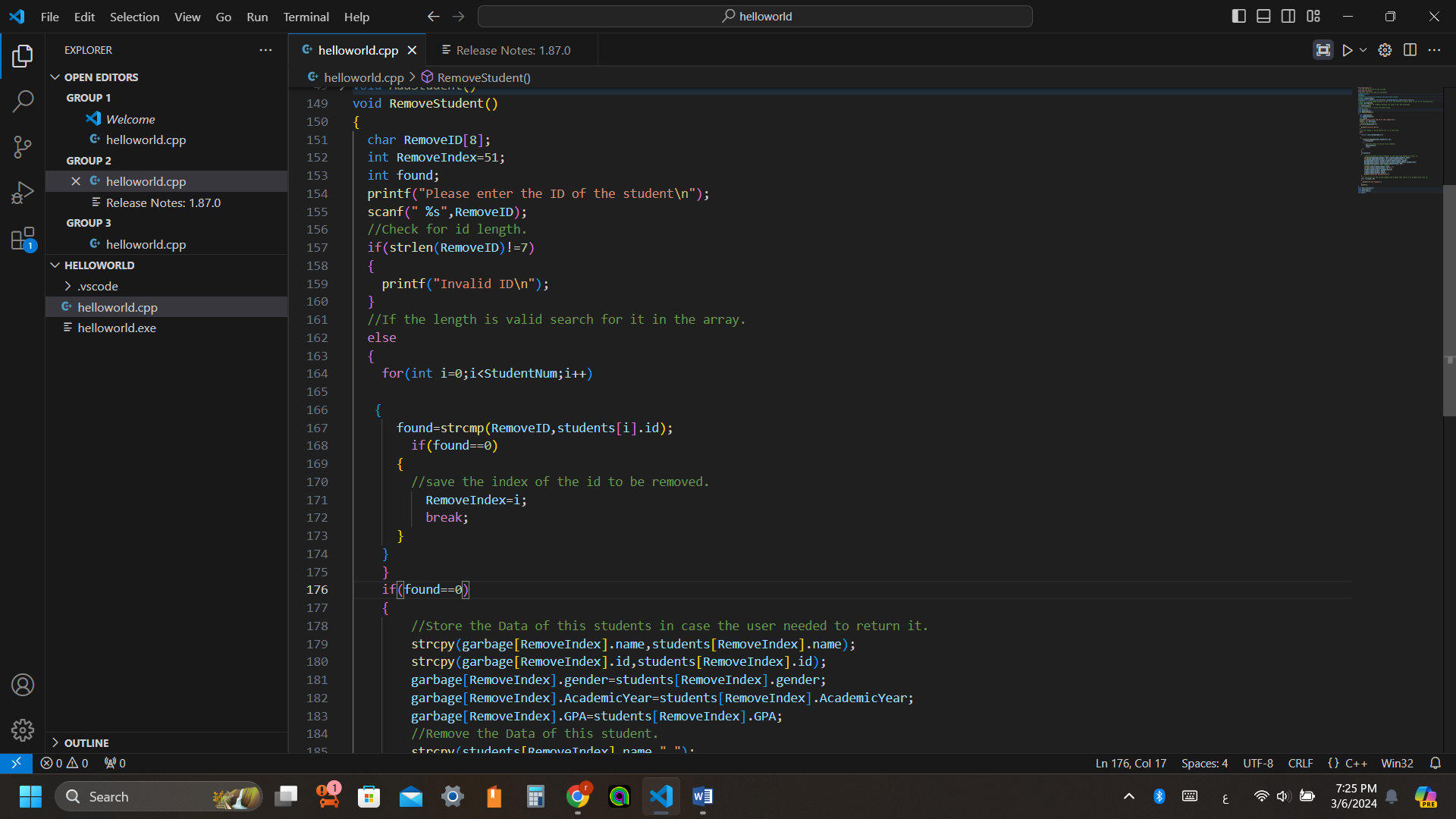
## Check gender

* Check if the user entered ‘F’ or ‘f’ for females, and ‘M’ or ’m’ for males.
* Then increment number of the students, to be added to the students’ data list.

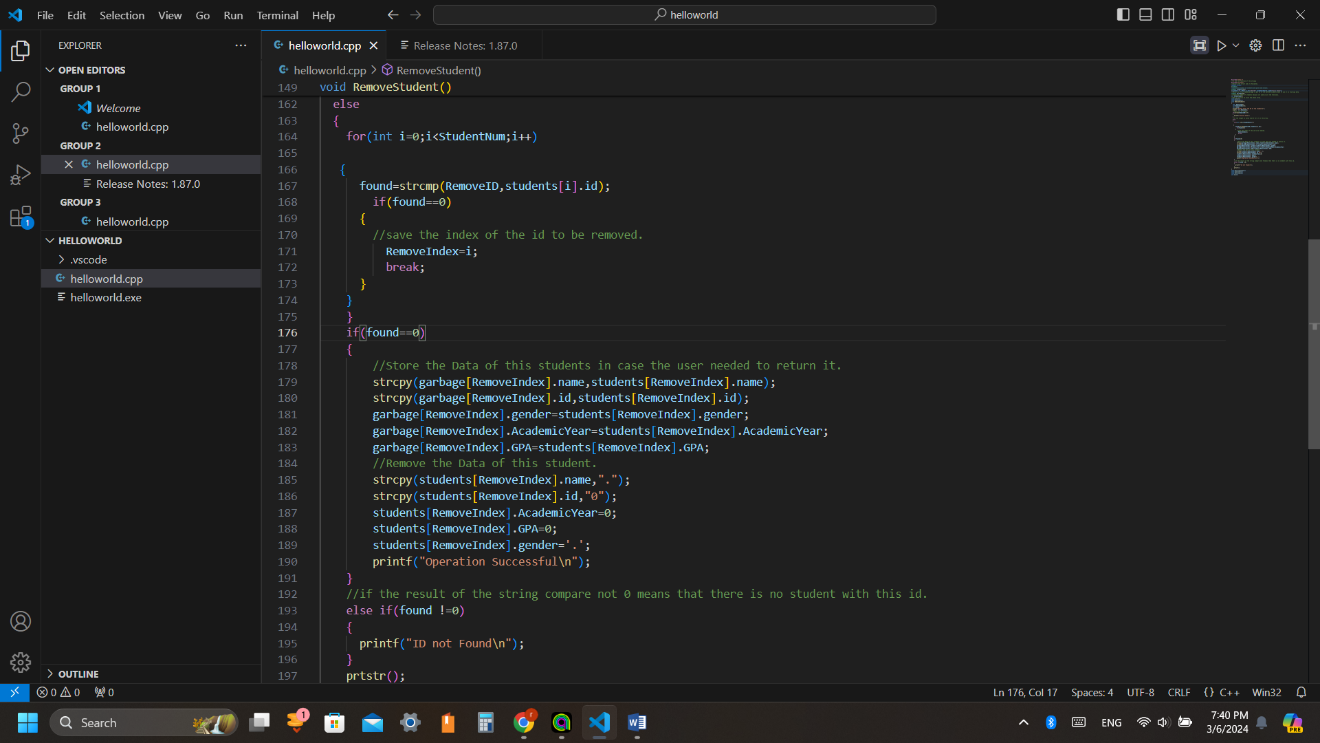
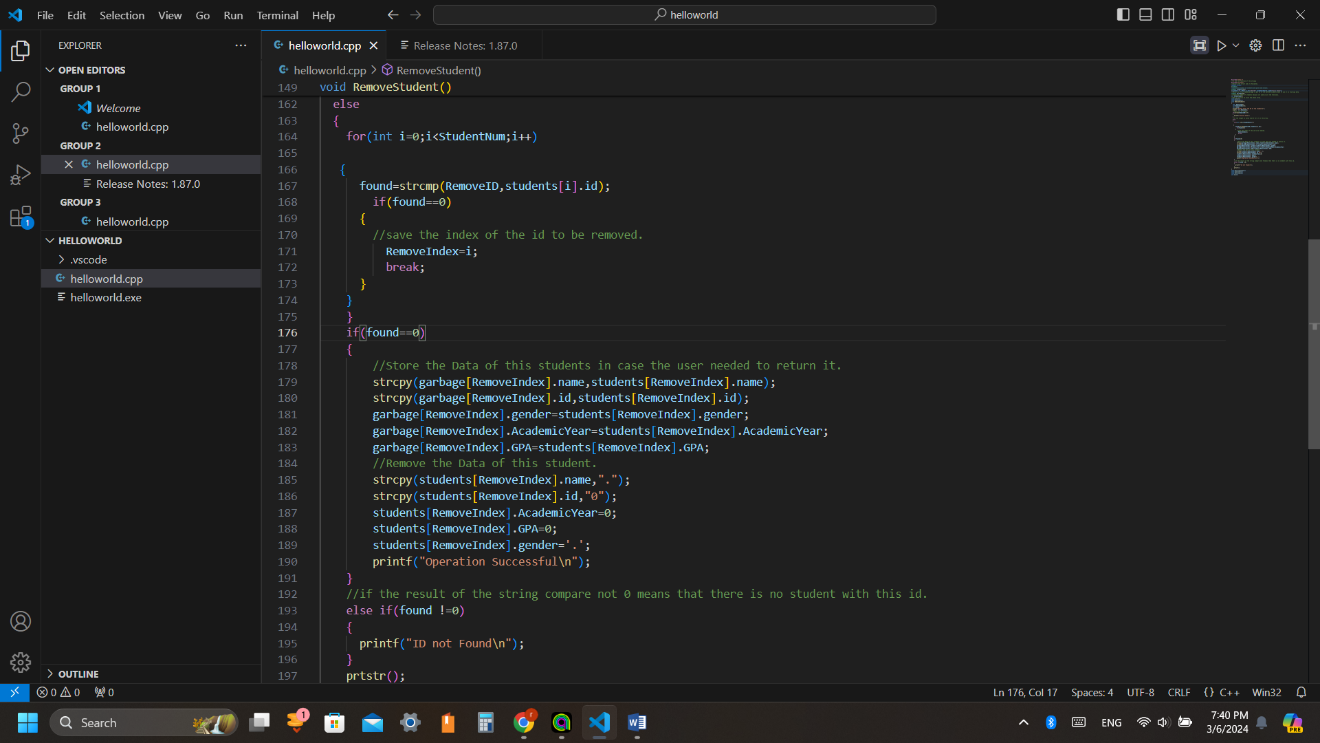
# Operation 3: **REMOVE**

## Check length:

* The system requires from the user to enter student ID that wanted to be removed.
* Then check ID length to assure its length not less or more than 7 digits.
* If not, print an error message “Invalid ID”.



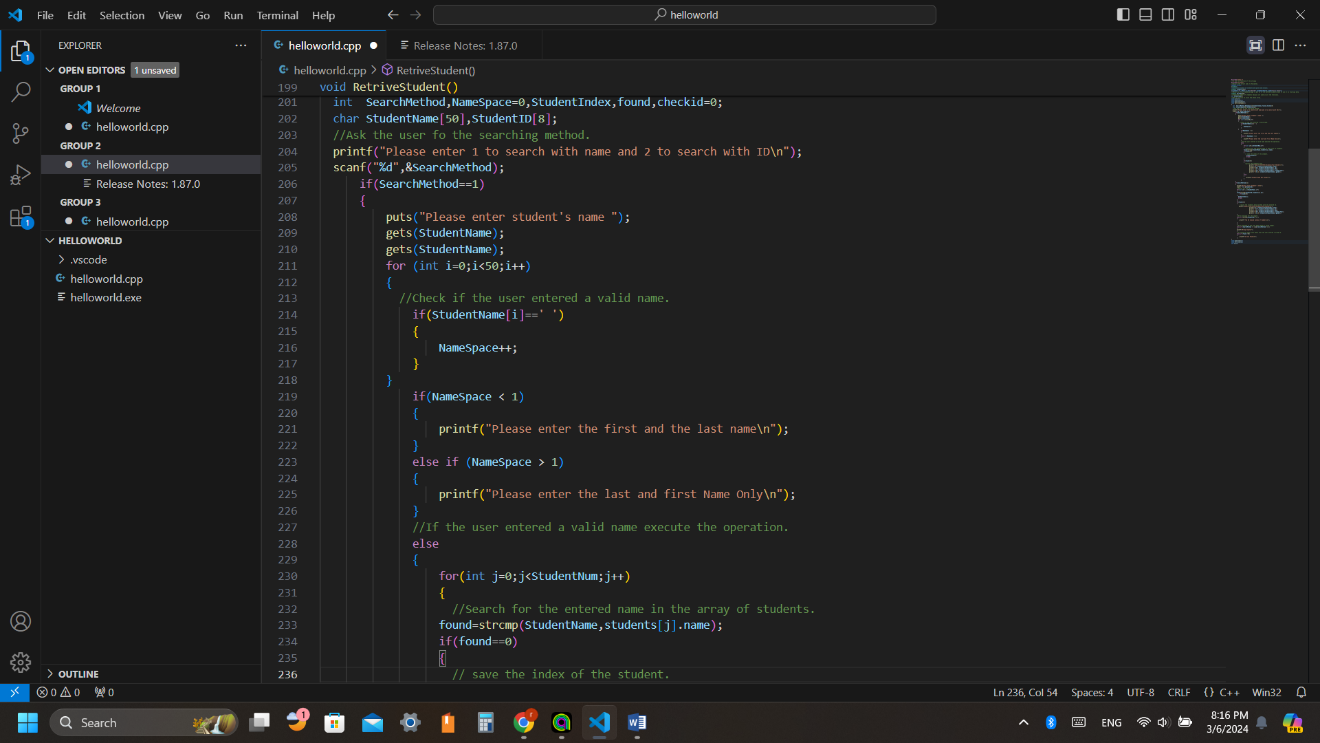
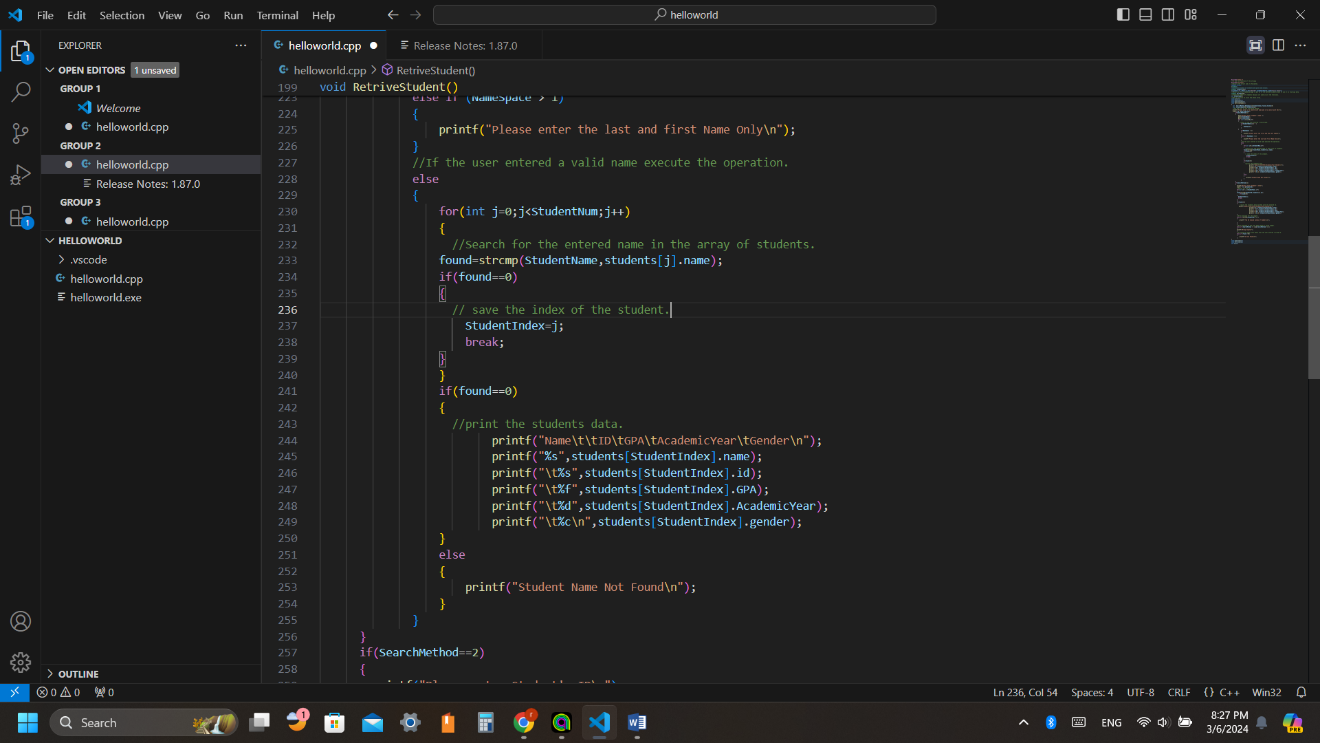
## Check validity:

* If the ID length=7 then start searching by the entered ID in the stored data (using found function).
* If ID founded (found==0), save the location of the removed student (RemoveIndex)
* Then store the data of the student in the garbage to restore it if needed.
* Then remove the data by setting its space with “0” or”.” which has length 1.
* If the ID is invalid in the stored data (found!=0), print an error message “ID not Found”.

# Operation 4: **Retrieve Student**

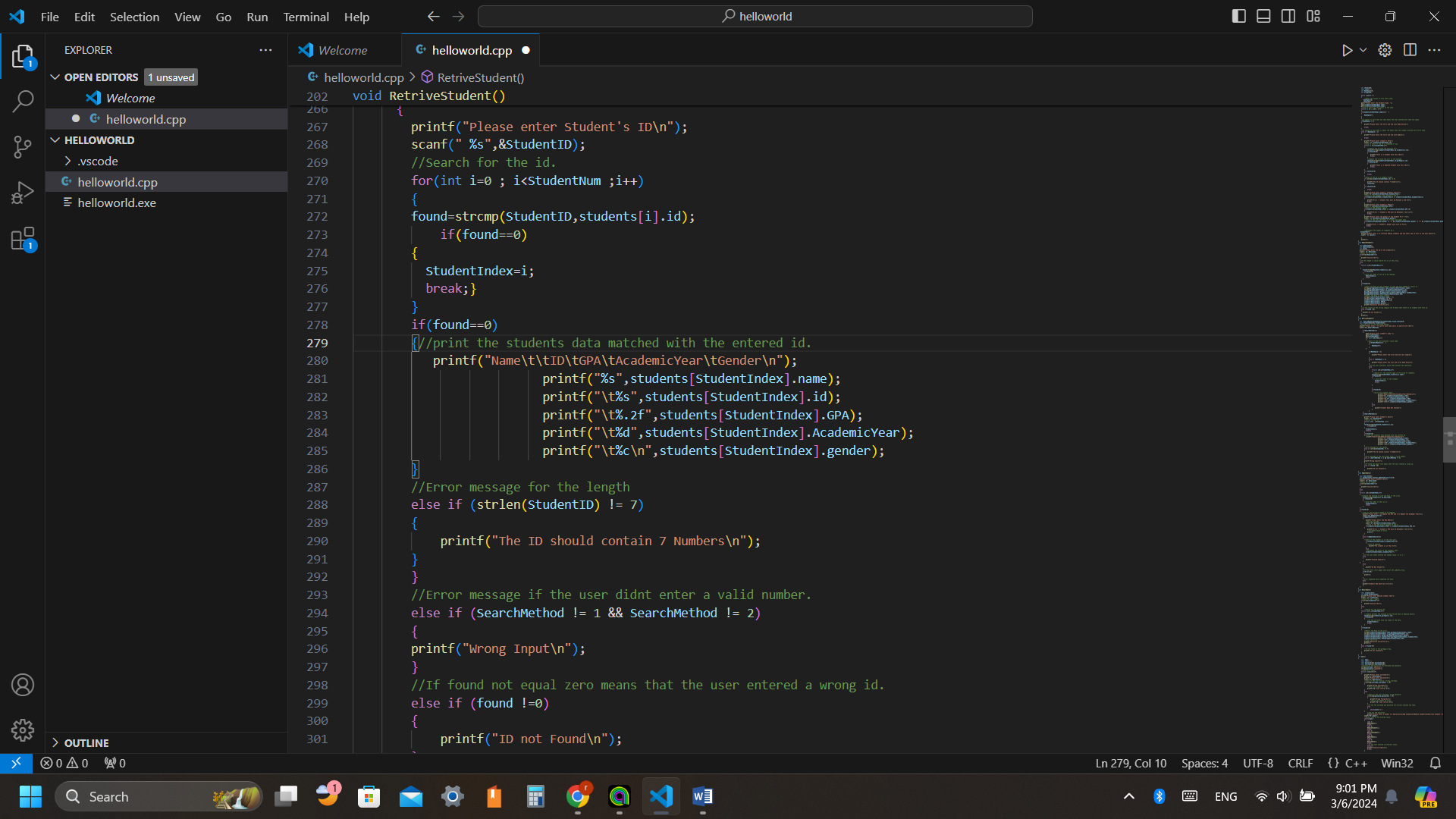
Retrieving a student will be done by one of two methods:

## Method 1: Search by name

* Check that the user entered the first and last name only as done in (Add Operation)
* Then search by name in the stored data using found function.
* If found, save the location, and print data of this location
* If not found, print an error message “Student name not Found”.

## Method 2: Search by ID

* Check that the user entered ID with length of 7 digits.
* Then search in the stored data, if founded, print the student data.
* If not found print an error message “ID not Found”.



# Operation 5: **Update**

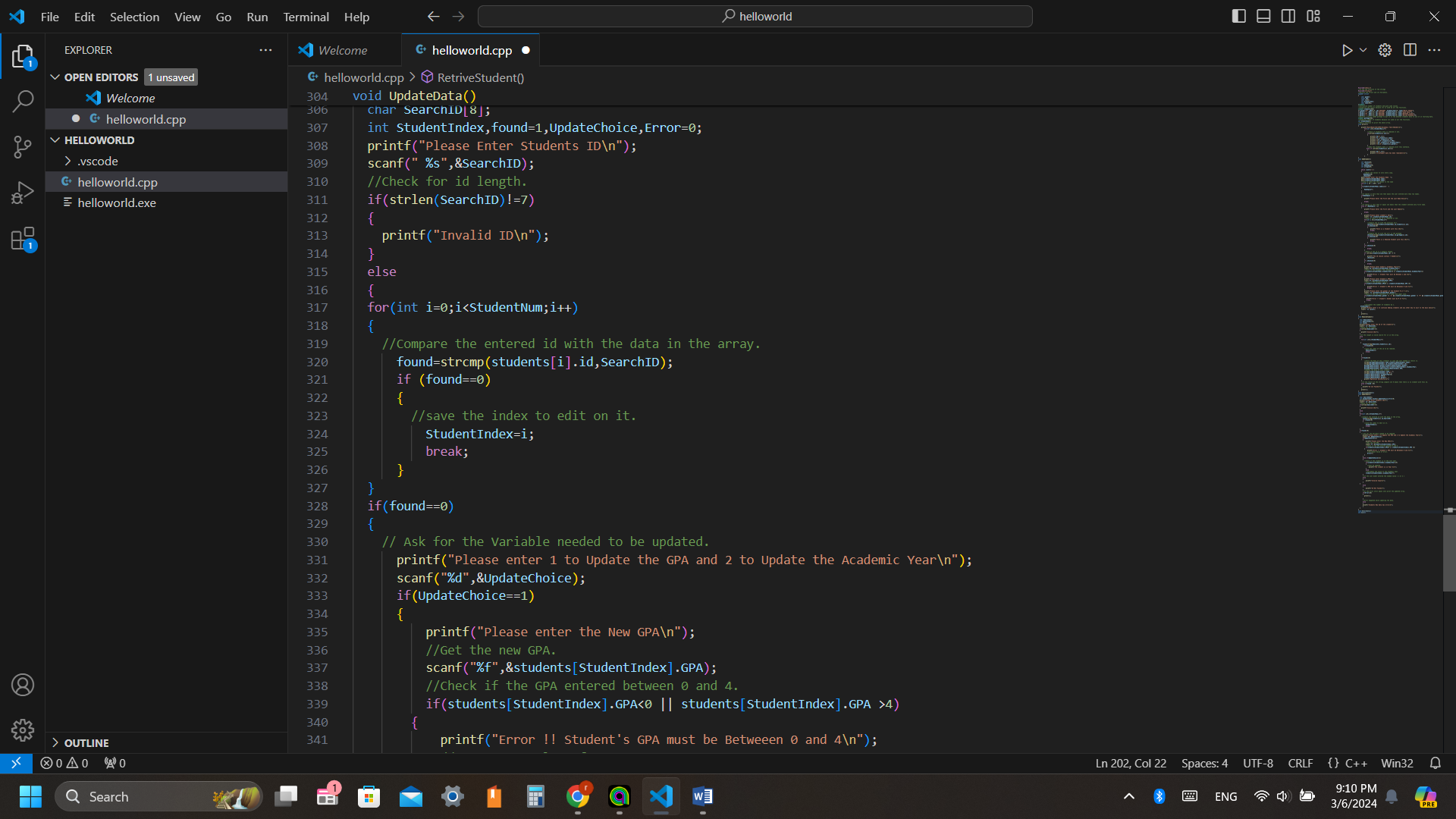
* Search for the student data that needed to be updated by ID.

1. Check ID length

* If valid, start to ask the user to enter a number to choose which data needed to be updated (1 to Update the GPA and 2 to update the Academic Year)

## **For updating GPA**:

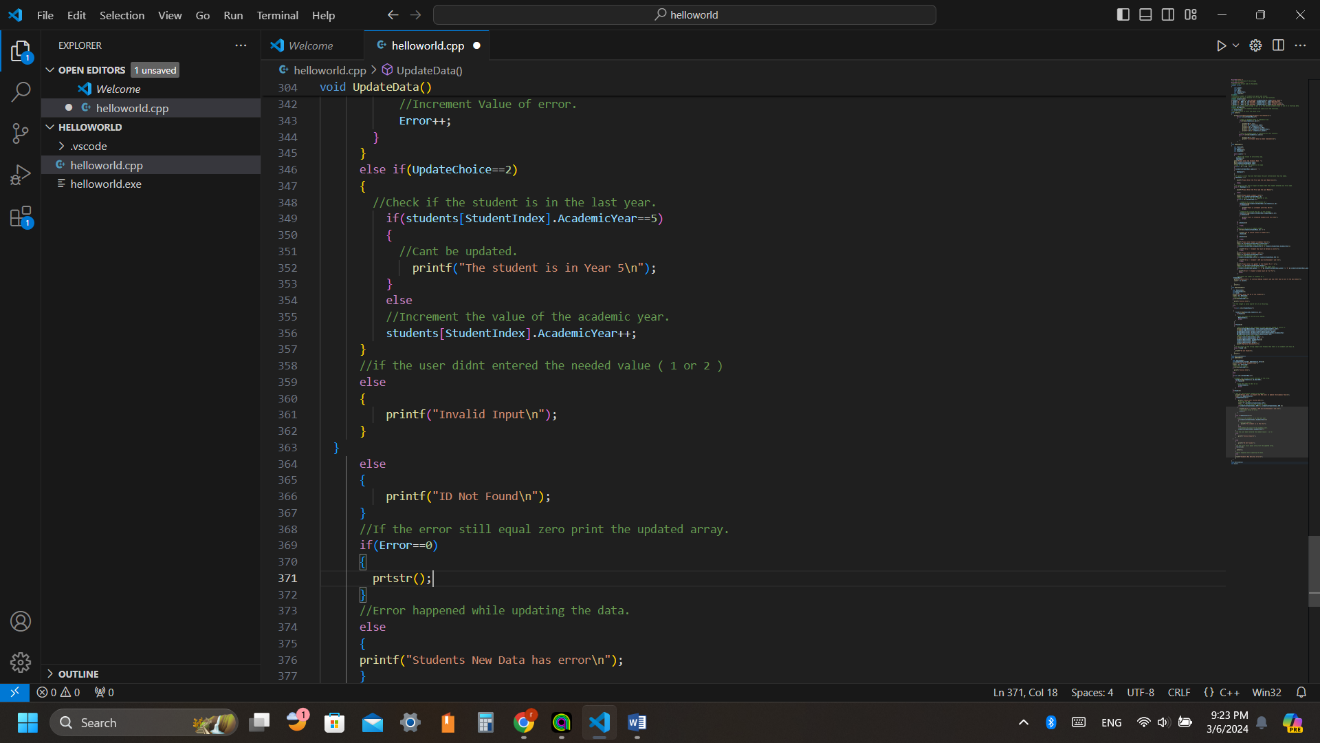
* Check that 0<= GPA <=4
* If not, print an error message “Error!! Student's GPA must be Between 0 and 4”.



## **For updating Academic Year**:

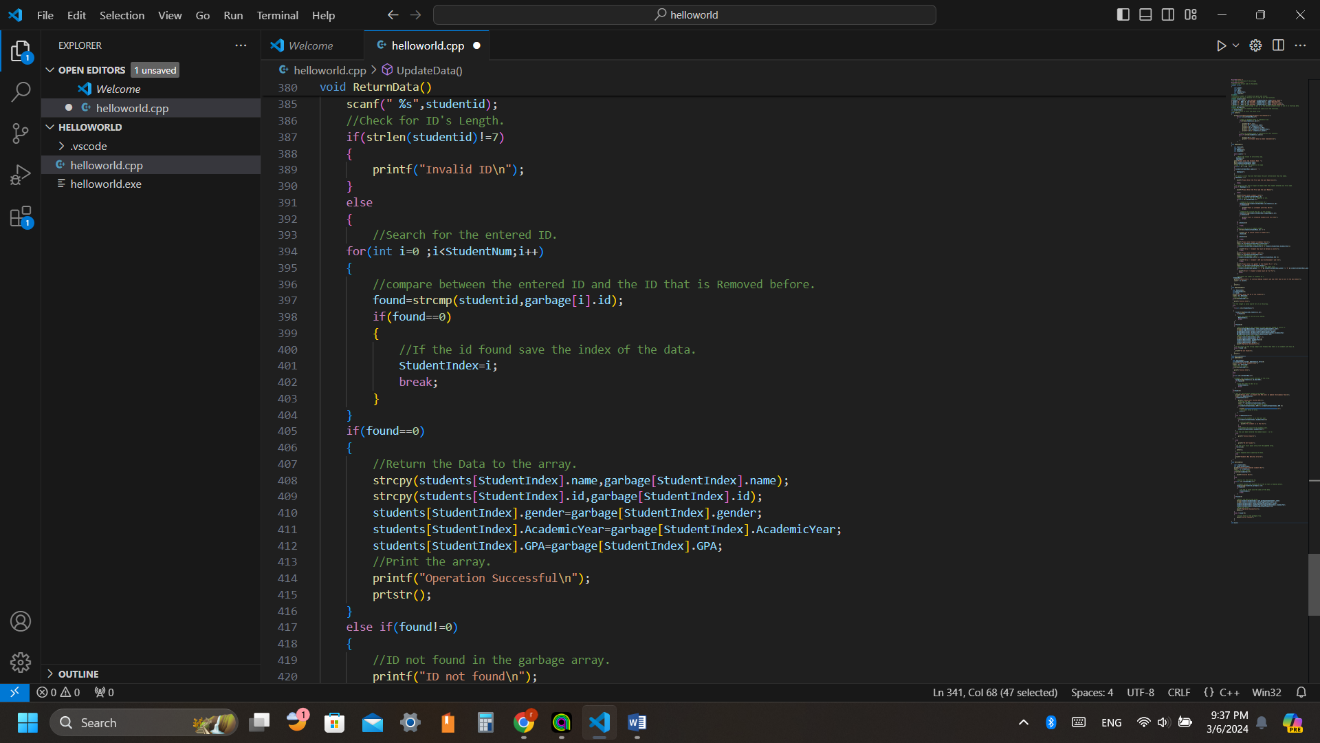
1. Check the academic year

* If the student in year 5 print a message that the student is in year 5.
* Other than, the academic year incremented by 1.



# Operation 6: **Return**

* To restore student’s data, search by ID
* Ask the user to enter the ID and check length to be =7
* Then start to search in the garbage array
* If found, save the student’s data location, then print the data.
* If not, print an error message “ID not found”



# Problems faced:

* The difficulty in using the strings as there is no string in the C language only some few functions.
* The number of Error checks were very big to do an efficient code.
* First time to do a big programming project.
* Declaring the structure and then edit on it (Globalize the struct).

# Conclusion:

In conclusion, this report outlines the design and operations of a Student Management System implemented in C using Visual Studio Code. The system ensures data integrity and user input validation through various error checks, providing functionalities for adding, removing, retrieving, updating, and returning student data. By implementing these operations, the system aims to provide a reliable platform for managing student information effectively.