Faculty of Engineering & Technology – Electrical & Computer Engineering Department

First Semester 2021 – 2022

Linux Laboratory – ENCS3130.

Python Project (Project 2)

Name: Ahmaide Al-Awawdah. **Name: Khalid Addabe**

ID: 1190823 **ID: 1190507**

Section: 1

Instructor: Dr. Mohammad Jubran

TA: Eng. Katy Sadi

Date: 26th December 2021 – 4th January 2022

1. Abstract

This is a python project using PyCharm to make a grading university system where the user can log in as either admin or student, with each having its own menu of options where the admin has six different options while the student only has two options.

The program also should check if there are previous registered students with their info (semesters, courses per semester) and stores them.

The program also checks the entered data and files if they are right or if they exist and it gives the user options to make wrong things right.

This project needs two files other than each student's file, and they are the students file (students.tx) that contains all the existing students ID's and each time a new student is added his/her ID is added to this file, the second one is the courses file (courses.txt) that contains all ENCS and ENEE available courses for computer engineering students with each credits of each course (1, 2, 3).

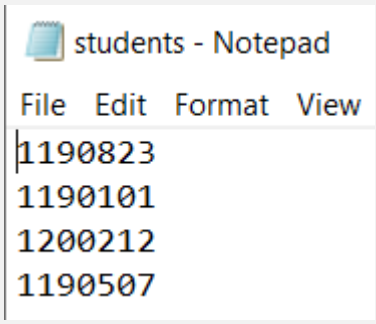


Figure 1: Students File

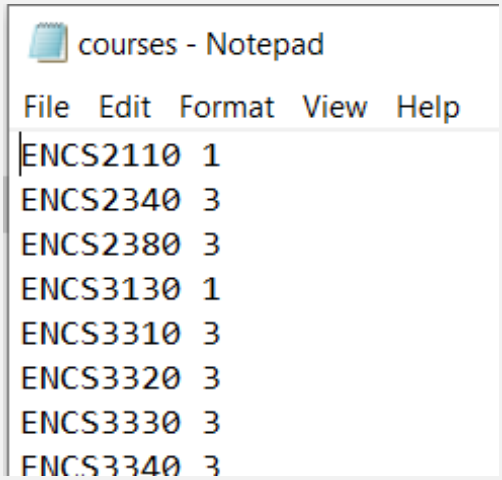


Figure 2: Courses File

❖Table of Content

2.Code Description

The code is included in the **Appendix 2**.

2.1. The Course Class

This class is for the course and it has three attributes (Course name (string), course grade (integer) initially zero, course credits (integer)) they are all private and in order for the user to get to them he should use the setter and getters.

A course Grade is initially zero that is also is used as a sign that the student didn't take the course where if the user is set to be failed in this course his grade will be (55-59).

2.2. The Student Class

This class is to initialize a student the constructure only takes two attributes where the class has five, the two attributes that the class takes are (student ID (integer), if file already exists (Boolean)) and the 5 student class attributes are:

- Student ID: the ID of the student (integer)
- Student File: the file of the student (id + ".txt") and could be existing before or needed to be made.
- Semesters: a list that has the all the semester that the student took.
- Subject per Semester: a list of lists each of these lists has the courses of the semester in the semesters list with the same index number.
- All Courses: a list of all the courses that the student could take with zero for the grade of the courses that the student didn't take yet.

The constructor gets if the students already has a file or not, if the student has a file it check if the file is available and if it wasn't it will give on option to add a new empty one, however if the file exists its data will be stored in the students list as each semester is added to the semesters list, and all this semesters courses are added to the list in the subject per semester list where this list will have the same index in the subject per semester list as the same as the semester in the list of semesters and the course grade is added in the list of all courses for this student, if the student is anew added to student then a new file will be created for this student.

The id and file name are private and the user should use getters in order to get to them. This class has a lot of functions for needs like (checking, getting index, adding, printing, and calculating) and these functions are:

2.2.1. The Getters

As said before the id and the file name are private variables where the user can only get to them via the getters.

2.2.2. Check New Semester

This function checks when adding a new semester in the form of (119, 120, …) with (Spring, fall, summer) then converting it in the form as “2019-2020/1” if this semester already exists for this student so it won’t be added as a new semester.

2.2.3. Semester Index

This function checks if the semester exists and if this student has it and if yes it returns its index.

2.2.4. Add Grade

This function adds a grade for a given course in a given semester, first it checks if the semester and course exist and then it adds this grade to this course in that semester.

2.2.5. Change Grade

This function changes the grade of the given index in the courses list with the given new grade.

2.2.6. Search for Semester of Courses

This function takes the course and checks if the student took it and then gives the semester that the student took this subject in.

2.2.7. Get Average

This function calculated the average of the student where it takes the courses that have a grade other than zero (55-99), where it takes the sum of the grade of each course timed to the number of its credits divided on the number of credit in order to get the average.

2.2.8. Get Credit Passes

This function calculates all the credits of the courses that the student took and got a grade higher than or equal to a 60 and returns the sum of the credits that the student passed.

2.2.9. Get Credit Taken

This function calculated all the credits of the courses that the student just took so either passed or failed the grade will be (55-99) and returns the sum of these credits.

2.2.10. Average per Semester

This function prints all the semesters of the student with hid average in each semester as the sum of each grade timed to its number of credit then this summation is divided to the number of credits.

2.2.11. Average and Credit For semester

This function takes the semester name, checks if the student took in this semester, then calculates the number of take credits and the average.

2.2.12. Remaining Courses

This function prints out all the courses that the student still didn’t take (grade = 0) with their credits and the summation of all the credits.

2.3. Used Data Structures

The used data structures in this project are all lists in order to store data from files to make it easier to get to them when it is needed, and they are:

2.3.1. The Courses List

This is list that stores all the courses that are read from the courses file “courses.txt” with their credits in order to make a copy of this list consisting all the courses with different objects that have the same course names and credit, and this copied list will be the all-courses list for each student.

2.3.2. The Students List

This is a list that stores all the students that are read from the students file “students.txt” in order to make them objects with each has its own file of information, if this file doesn’t already exist, and each time the user adds a new student the object will be added to this list as the student info is added to that student’s file.

2.3.3. The Semesters List

This list stores all the possible semesters that the stored students have as each semester need at least one student to be registered in it to be included in this list.

2.4. Functions

2.4.1. Make Courses List

This function makes and returns a copy of the students list in order for a student to have this list.

2.4.2. Add to Semester List

This function adds a new semester to the semester list.

2.4.3. Check ID

This is probably the most used function in this project where calling it in order to ask the user to enter a student id, it checks if the id is real then it checks if it exists for a student, if not it gives a user a choice to re-enter the id it will return the id with Boolean value to tell if the id is valid or not.

2.4.4. Enter ID and Check if It Exists

This function checks if the id is unique or not, if not it gives the user an option to re-enter the id, it will return the id with a Boolean value to tell if the id works or not.

2.4.5. Check Semester Info

This function takes a student and asks the user to enter a semester in the year form (119, 120, ...) and a semester (spring, summer, fall) and checks if this semester is valid or if it already exists, and asks the user to re-enter a semester, it will return the semester string with a Boolean value that tells if the semester info works or not.

2.4.6. Find Index and Check

This function searches for the index of a student in the students list and returns his index and if this student exists or not.

2.4.7. Check and Find Course

This function searches for the index of a course in the courses list, and checks if this course exists or not.

2.4.8. Replace in File

This function takes the student, the semester, the course, and the new grade, it takes the semester index as the row which is also the index of the list of courses of that semester in the subject per semester list of that student the course index in the inner list will be the column in the file of the student that contains the old grade that needs to be changes then it takes the lines of the file, the line of that semester is split into a list of characters where the numbers of the old grad are replaced with the new ones, then the list is fixed back into a string of that semester then all the lines are re-entered into the file again as everything as same as before except for the new grade.

2.4.9. Semester Sort

This function simply sorts the semester list using the selection sort, so that the list is sorted from the prior to the least semester in order to be printed in that sort.

2.5. The Main

After the project reads and stores data from the students and courses file and the previous existing students files, the system asks the user weather he is a student or an admin, this will only work if the previous courses file exists, so the courses are stored in the courses list and the previous students are saved in students list.

The user has to be one of either two admin or a student, the student must enter his id in order to get the info that he needs and the system checks if this id is available or not, the admin has six menu options while the student has only two out of the six, these six choices are:

2.5.1. Adding a student

This option is only available for the admin, where the admin enters a new student id, the system checks using the enter id and check if it exists function in order to make sure that the id is unique, if the id is unique and that ID isn't already used the system makes a new student object with a new student file and prints in it for the first line ("Year/Semester ; Courses with Grades"), with courses grades initialized as zero.

2.5.2. Adding a Semester of Courses

This option is also only available for the admin, where the admin enters the student's id and checks if the student exists using find index and check function, the system asks the admin user to enter a new semester year In the form of (119, 120, …) and the semester (fall, spring, summer), and checks if the user entered a right existing semester, then a loop will go on asking the user admin to enter courses for the student until the user types "exit" and while that the system checks if that course exists and if the grade of that course is less than 55, the students grade will be 55, after all the info is ready and added to lists of semester and courses of that student, all new information of that semester consisting all its courses with their grades are added to the file.

2.5.3. Change a Grade

This option just for the admin, where the admin enters the students id that has that grade to be changed, the system checks if the id is right and exists, it also does the same thing for the course name, it also checks if the student did already take this course and his grade is not zero, if all that was right the system looks for the index of the course and then changes its grade to the new entered one, then the replace in file function changes the grade in the student file with the new grade as the semester of that course is also searched for in order for the grade to be changed in its row.

2.5.4. Student Statistics

This option is available for both student and admin, where admin searches for a student to get his statistics, and the student user gets his own statistics, as it prints the number of taken credits by that student, and the number of passed credits, then it prints out the remaining courses for the student, with the remaining credits, and then it print the average for each semester that student took, and in the last it prints the overall average for that student

2.5.5. Global Statistics

This option is available for both student and admin, as it prints the overall students average, then it prints the overall average for each of the available semesters from the available info, and then it plots a histogram of the average of students (60s, 70s, 80s, 90s) using the “matplot” library in python as shown in figure 3.

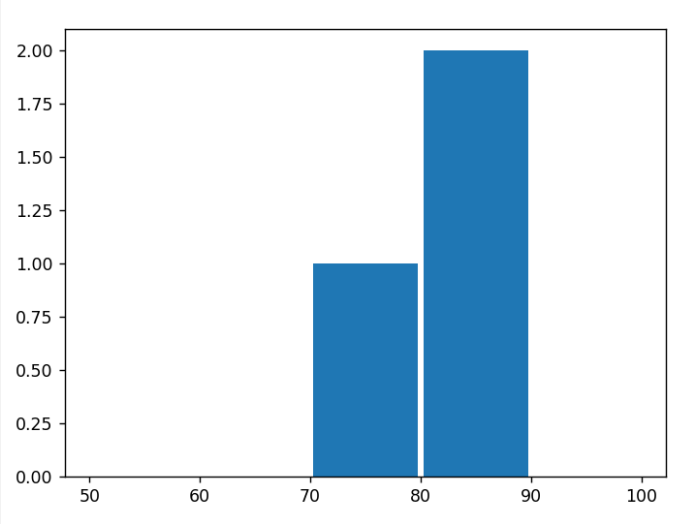


Figure 3: Example of histogram of averages of students

2.5.6. Searching by average or credits.

This option is only available for the admin, where it gives the admin a choice to print the students weather by the average or by the number of passed credits, the user then gives the number of credits or average to be searched for then the user chooses weather if he wants to print the students who got higher or less than this average or passed more or less than this number of credits.

3. Test Cases

For the test cases, here are the available files before running the program.

students - Notepad

File Edit Format View

```
1190823
1190101
1200212
1190507
1211010
```

courses - Notepad

File Edit Format Vie

```
ENCS2110 1
ENCS2340 3
ENCS2380 3
ENCS3130 1
```

1190823 - Notepad

File Edit Format View Help

```
Year/Semester ; Courses with Grades
2020-2021/1 ; ENCS2340 92, ENEE2307 87
2020-2021/2 ; ENCS2110 88, ENCS2380 90, ENEE2304 86, ENEE2312 73
```

1190101 - Notepad

File Edit Format View Help

```
Year/Semester ; Courses with Grades
2021-2022/1 ; ENCS2340 92, ENEE2304 81, ENEE2307 87, ENCS3130 55
2021-2022/2 ; ENCS2380 99, ENEE2312 96, ENCS2110 90
2021-2022/3 ; ENEE2360 61, ENCS4110 94
```

1200212 - Notepad

File Edit Format View Help

```
Year/Semester ; Courses with Grades
```

1190507 - Notepad

File Edit Format View Help

```
Year/Semester ; Courses with Grades
```

1211010 - Notepad

File Edit Format View Help

```
Year/Semester ; Courses with Grades
2021-2022/1 ; ENCS2340 71, ENEE2360 55, ENEE3309 96
```

Info: the courses file info is included in **Appendix 1**.

3.1. Test Case 1

In this case a semester will be added to 1190507, with some errors to see how to system must fix them.

```
     Welcome to Student Record System
*****************************************

Are you an Admin or a Student: AdmIN

     THE MENU
******************
1- Add a New Record
2- Add a New Semester for Student
3- Update Grade
4- Get Student Statistics
5- Get Global Statistics
6- Search Based on Credits or Grades
others - Exit
-----------------------------------
Enter your choice here: 2
Enter the ID of the Student: 119990
The id isn't current,
enter 1 if you want to change it: 1
Enter ID: 1199990
That student doesn't exist!
Enter 1 if you want to try again: 1
```

```
Enter the ID of the Student: 1190507
Enter the year in form of(118, 119, 120...): 120
Enter semester (1: for fall, 2: for spring, 3: for summer): 1
Enter a new course name or type exit to leave: ENCS2340
Enter the Grade: 95
Enter a new course name or type exit to leave: COMP2421
Enter the Grade: 90
The course isn't available!
Enter a new course name or type exit to leave: ENEE2360
Enter the Grade: 88
Enter a new course name or type exit to leave: exit

     THE MENU
******************
1- Add a New Record
2- Add a New Semester for Student
3- Update Grade
4- Get Student Statistics
5- Get Global Statistics
6- Search Based on Credits or Grades
others - Exit
-----------------------------------
Enter your choice here: 7
Program is Exiting.........

Process finished with exit code 0
```

The figure below shows the changes in the student 1190507 file

```
1190507 - Notepad
File  Edit  Format  View  Help
Year/Semester ; Courses with Grades
2020-2021/1 ; ENCS2340 95, ENEE2360 88
```

3.2. Test Case 2

In this case the student 1190101 grade of ENCS3130 will be changed from 55 to 77 and adding a new student with id 1190000.

```
     Welcome to Student Record System
****************************************

Are you an Admin or a Student: admin

     THE MENU
*****************
1- Add a New Record
2- Add a New Semester for Student
3- Update Grade
4- Get Student Statistics
5- Get Global Statistics
6- Search Based on Credits or Grades
others - Exit
---------------------------------
Enter your choice here: 3
Enter the ID of the Student: 1190101
Enter Course ID: ENCS3130
Enter new Grade: 77
```

```
     THE MENU
*****************
1- Add a New Record
2- Add a New Semester for Student
3- Update Grade
4- Get Student Statistics
5- Get Global Statistics
6- Search Based on Credits or Grades
others - Exit
---------------------------------
Enter your choice here: 1
Enter the ID of the New Student: 1190000

     THE MENU
```

The changes in the files:

1190101 - Notepad

File Edit Format View Help

```
Year/Semester ; Courses with Grades
2021-2022/1 ; ENCS2340 92, ENEE2304 81, ENEE2307 87, ENCS3130 77
```

1190000 - Notepad

File Edit Format View Help

```
Year/Semester ; Courses with Grades
```

students - Notepad

File Edit Format View

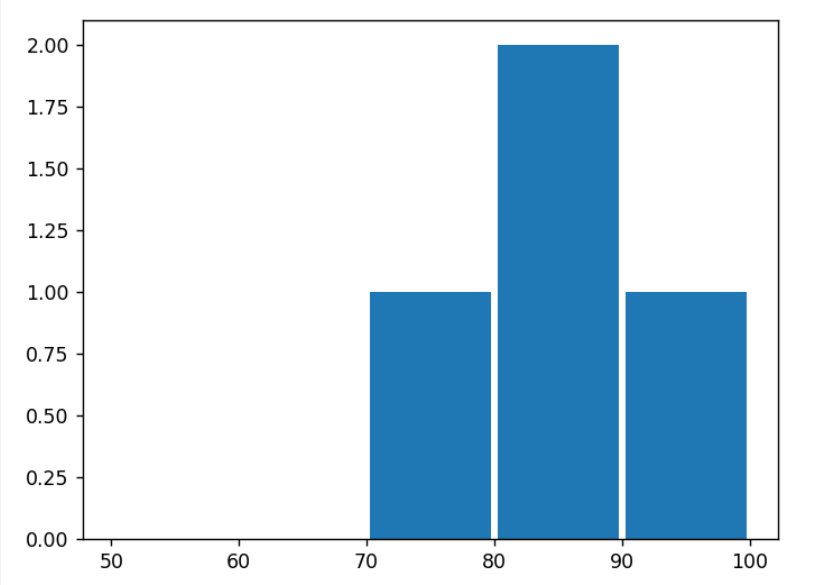
```
1190823
1190101
1200212
1190507
1211010
1190000
```

3.3. Test Case 3

In this case the student 1190823 in going to register and print his statistics and the global statistics.

```
     Welcome to Student Record System
****************************************

Are you an Admin or a Student: student
Enter the ID of the Student: 1190823

     THE MENU
******************
1- Get Student Statistics
2- Get Global Statistics
others - Exit
-----------------------------------
Enter your choice here: 1
Taken credits:  16
Passed credits:  16


Remaining courses:
ENCS3130 , ENCS3310 , ENCS3320 , ENCS3330 ,
ENCS3340 , ENCS3390 , ENCS4110 , ENCS4130 ,
ENCS4210 , ENCS4300 , ENCS4310 , ENCS4320 ,
ENCS4330 , ENCS4370 , ENCS4380 , ENCS5140 ,
ENCS5150 , ENCS5200 , ENCS5300 , ENEE2103 ,
ENEE2360 , ENEE3309 , ENEE4113 ,

Remaining Credits:  53
```

```
Semester / Average
2020-2021/1 / 89.5
2020-2021/2 / 83.5

Overall Average:  85.75

     THE MENU
******************
1- Get Student Statistics
2- Get Global Statistics
others - Exit
-----------------------------------
Enter your choice here: 2

Overall students average : 84.53846153846153

Semester, overall average, average credit
2020-2021/1: 90.5, 6.0
2020-2021/2: 83.5, 10.0
2021-2022/1: 80.15789473684211, 9.5
2021-2022/2: 96.42857142857143, 7.0
2021-2022/3: 69.25, 4.0

The histogram of the averages should be displayed!
```



3.4. Test Case 4

In this case the file of the student 1200212 is removed yet the id is still mentioned in the students file, and the user will enter something other than student or admin.

```
The student with the ID:  1200212 , doesn't have a file!
Enter 1 to add a new file: 1

     Welcome to Student Record System
*****************************************

Are you an Admin or a Student: else101

That's not a valid option, exiting.........

Process finished with exit code 0
```

3.5. Test Case 5

In this case the user in going to search the students who have an average less than 87.

```
     Welcome to Student Record System
*****************************************

Are you an Admin or a Student: admin

     THE MENU
******************
1- Add a New Record
2- Add a New Semester for Student
3- Update Grade
4- Get Student Statistics
5- Get Global Statistics
6- Search Based on Credits or Grades
others - Exit
-----------------------------------
Enter your choice here: 6
enter 1 to search by credits, 2 to search by average: 2
Enter the average to search for: 87
Enter 1 to searched by more, 2 to search by less than: 2
Student ID, Average
1190823    ,  85.75
1190101    ,  86.14285714285714
1200212    ,  0
1211010    ,  74.0
1190000    ,  0
```

4. Conclusion

In conclusion we learned how to make python programs using object oriented, functions, exception handling, file read and write.

We made a system that can enter, change, print, and replace students' info (semesters, courses, grades) as needed in files too, with covering error cases as much as we could with giving the user the opportunity to fix the wrong data that they type in and his missing information too.

Python is different world from C and java, from this project we think that the splitting into lists functions is one of the best features that makes it special programming language where it can be a first choice in file reading and string splitting projects in the future.

5. Appendix

5.1. Appendix 1

ENCS2110 1
ENCS2340 3
ENCS2380 3
ENCS3130 1
ENCS3310 3
ENCS3320 3
ENCS3330 3
ENCS3340 3
ENCS3390 3
ENCS4110 1
ENCS4130 1
ENCS4210 2
ENCS4300 3
ENCS4310 3
ENCS4320 3
ENCS4330 3
ENCS4370 3
ENCS4380 3
ENCS5140 1
ENCS5150 1
ENCS5200 2
ENCS5300 3
ENEE2103 1
ENEE2304 3
ENEE2307 3
ENEE2312 3
ENEE2360 3
ENEE3309 3
ENEE4113 1

5.2. Appendix 2

```
import matplotlib.pyplot as plt
# ============ Class: Courses =============================================================
class Course:
    # This class will have the course with its grade and credits
    def __init__(self, name, credit):
        self.__name = name
        self.__credit = credit
        self.__grade = 0    # grade 0 means that the student didn't take this course yet

    def getName(self):
        return self.__name

    def getGrade(self):
        return self.__grade

    def getCredit(self):
        return self.__credit

    def setGrade(self, grade):
        self.__grade = grade



#^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ Initializing all Courses & semester list ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
try:
    couFile = open("courses.txt", "r")     # The system will check if the file exists
    courseList = couFile.readlines()       # Then it saves all the available courses in
    for i in range(0, len(courseList)):    # a list called "courseList" with all grades = 0
        courseInfo = courseList[i].split(" ")
        courseObj = Course(courseInfo[0], int(courseInfo[1]))
        courseList[i] = courseObj
    couFile.close()
```

```
    coursesExist = True
except FileNotFoundError:      # If the file doesn't exist the system will ask the user to make the file
    print("The courses file doesn't exist Please give in the file!")
    courseList = []
    coursesExist = False
semesterList = []


#------------- Function: New Courses List --------------------------------------------------------
def MakeCoursesList():
    newlist=[]                 # This function will make a copied courses list with different objects
    for course in courseList:   # Of each course for each student, with initial grade = 0
        CopiedCourse = Course(course.getName(), course.getCredit())
        newlist.append(CopiedCourse)
    return newlist

#------------------ Function: Add to Semester List ------------------------------------------------
def addToSemesterList(semester):
    notThere = True            # This function is to save all the semesters of all students
    for sem in semesterList:    # and it also checks that the semester already exists
        if semester == sem:
            notThere = False
    if notThere:
        semesterList.append(semester)


#^^^^^^^^^^^^^ Initializing Previous Students List Part 1 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
try:
    stuFile = open("students.txt", "r+")    # This function makes a list of all the students from the file
    studentList = stuFile.readlines()       # and makes sure that the students file exists
    stuFile.close()                         # It also gives the option to add a file if it doesn't exist
    PrevStudExist = True
except FileNotFoundError:
```

```
    print("No students file, so a new one will be created")
    with open("students.txt", 'a+') as fi:
        fi.write("")
        fi.close()
    PrevStudExist = False
    studentList = []


#============= Class: Student ==================================================================
class Student:
    # the student attributes are the id, file name, semesters list, all courses list, and course per semester list
    def __init__(self, id, FileExist):
        # File exists is a boolean to determin the condition of weather the student is new or already has a file
        self.__id = id                    # The id of the student is an integer
        self.__file = (str(id) + ".txt")    # The name of the student file ( ID + .txt )
        self.semesters = []               # A list of all the semesters the student taken
        self.subjPerSem = []              # A list of the lists of courses of each semester
        self.allCourses = MakeCoursesList() # A list of all the courses that the student must take
        if FileExist:   # The case of a previous student who already has a file
            try:       # the exception in case it was said that the student has a file but actually don't
                f = open(self.__file, "r+") # Storing students data after they are redden from his file
                fileLines = f.readlines()   # Split the file in a list for each line
                if len(fileLines) > 1:      # Start from the second line because the first is to describe the columns
                    for i in range(1, len(fileLines)):
                        j = i - 1   # the line number is a head of the index by 1
                        lineSplit = fileLines[i].split(" ; ")   # To split the semester from the courses
                        self.semesters.append(lineSplit[0])     # Add the semester to the semesters of the student
                        addToSemesterList(lineSplit[0])
                        self.subjPerSem.append([])              # add a new list of courses to the subject per semester
                        if (len(fileLines[i]) > 13):            # To check that there are courses at this semester
                            subjSplit = lineSplit[1].split(", ")    # Courses split
                            for subjStr in subjSplit:
                                subjInfo = subjStr.split(" ")       # Split course ID from it's grade
```

```
                    for course in self.allCourses:
                        if course.getName() == subjInfo[0]: # Add the grade if its greater than previous
                            gra = int(subjInfo[1])
                            if gra > course.getGrade():
                                course.setGrade(int(subjInfo[1]))
                                self.subjPerSem[j].append(course)
            f.close()
        except FileNotFoundError:   # In case there is no file, the system gives the user option to add file
            print("The student with the ID: ", self.__id, ", doesn't have a file!")
            choose = input("Enter 1 to add a new file: ")
            if len(choose)>0:
                if choose[0]=='1':
                    with open(self.__file, 'a+') as fi:
                        fi.write("Year/Semester ; Courses with Grades")
                        fi.close()
                else:
                    print("student won't be added!")
            else:
                print("student won't be added!")
    else:   # The case of the student is new and doesn't have a file so it can be added
        with open(self.__file, 'a+') as fi:
            fi.write("Year/Semester ; Courses with Grades")
            fi.close()

    def getId(self):
        return self.__id

    def getFile(self):
        return self.__file

    def checkNewSemester(self, year, sem):  # Checking if this student has the semester in term of (119,120...)
        semester = (str(year+1900) + "-" + str(year+1901) + "/" + str(sem))
        s = True
```

```
        for se in self.semesters:
            if semester == se:
                s = False
        return s, semester

    def semesterIndex(self, semester):  # Checks that the semester exists and gets its index
        s = False
        index = len(self.semesters)
        for se in self.semesters:
            if semester == se:
                s = True
                index = self.semesters.index(se)
        return s, index

    def addGrade(self, course, semester, grade):    # Add a new grade for the student
        ind = len(self.semesters)                   # and checks if the course exists to be added
        courseP = 0
        courseFound = False
        semesterFound = False
        for s in self.semesters:
            if semester == s:
                ind = self.semesters.index(s)
                semesterFound = True
        for c in self.allCourses:
            if course[0:8] == c.getName():
                courseP = c
                courseFound = True
        if semesterFound and courseFound:
            if grade > courseP.getGrade():
                courseP.setGrade(grade)
            self.subjPerSem[ind].append(courseP)
        return (courseFound), courseP
```

```
    def changeGrade(self, index, newGrade):     # Changes the grade by its index
        cour = self.allCourses[index]
        cour.setGrade(newGrade)

    def searchForSemesterOfCourse(self, cou):   # Get the semester that the student took that course in
        index = 0
        index2 = 0
        for sem in self.subjPerSem:
            for subj in sem:
                if cou.getName == subj.getName:
                    index = self.subjPerSem.index(sem)
                    index2 = sem.index(subj)
        return index, index2

    def getAvg(self):   # Calculates the total average of the student
        sum=0
        cred=0
        for cou in self.allCourses:
            if cou.getGrade()>=55:
                sum+=(cou.getGrade() * cou.getCredit())
                cred+=cou.getCredit()
        if(cred!=0):
            average = float(sum)/float(cred)
            return average
        else:
            return 0

    def getCredPass(self):  #Gets the total passed credits of the student
        cred = 0
        for cou in self.allCourses:
            if cou.getGrade()>=60:
                cred+=cou.getCredit()
        return cred
```

```
    def getCredTaken(self): #Gets the total registered credits for the student
        cred = 0
        for cou in self.allCourses:
            if cou.getGrade()>=55:
                cred+=cou.getCredit()
        return cred

    def avgPerSem(self):    # Prints the student's average in some certain semester
        print("\nSemester / Average ")
        for numSem in self.semesters:
            index = self.semesters.index(numSem)
            s = numSem + " / "
            graSum = 0
            credSum = 0
            for matNum in self.subjPerSem[index]:
                graSum+=(matNum.getGrade()*matNum.getCredit())
                credSum+= matNum.getCredit()
            i = 0
            if (credSum != 0):
                i = float(graSum)/float(credSum)
            s = s + str(i)
            print(s)

    def avgAndCredForSem(self, semName):    # Gets both average and credits for a semester
        check, index = self.semesterIndex(semName)
        avg =0
        cr = 0
        if check:
            sum=0
            for cou in self.subjPerSem[index]:
                sum+= (cou.getGrade() * cou.getCredit())
                cr+= cou.getCredit()
```

```
            if cr != 0:
                avg = float(sum) / float(cr)
        return check, avg, cr


    def remainingCourses(self):    # Prints the remaining courses for the student
        print("Remaining courses: ")
        counter = 0
        for c in self.allCourses:
            if c.getGrade()<60:
                #print(c.getName(),",", end=" ")
                if(counter %4 == 3):
                    print(c.getName(),", ")
                else:
                    print(c.getName(), ",", end=" ")
                counter += 1
        remInt = 69 - self.getCredPass()
        print("\n\nRemaining Credits: ", str(remInt))



#^^^^^^^^^^^^^^^^^^^^^^ Initializing Previous Students List Part 2 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
if PrevStudExist:
    counter = 0    # Giving the students ID's in the file objects with all their info in their file
    for student in studentList:
        fName = (studentList[counter] + ".txt")
        intId = int(studentList[counter])
        stud = Student(intId, True)
        studentList[counter] = stud
        counter = counter + 1



#---------------- Function: ID Check -----------------------------------------------------------
def CheckId(id):
    s = False      # This function is to make the user enter a student id and check if it is exists
```

```
    choose = 0      # And it keeps giving the user chances in option if they entered wrong data
    while (not s) and (choose==0):
        if  id.isdecimal() and len(id) == 7:
            s = True
        else:
            choise = input("The id isn't current,\nenter 1 if you want to change it: ")
            if choise[0] == '1':
                choose = 0
                id = input("Enter ID: ")
            else:
                choose =1
    return s, id



#--------------- Function: New ID and Check If it is unique -------------------------------------------
def EnterIDAndCheckIFItAlreadyExists():
    lock = True          # This function checks if the id is unique
    check = False        # And it keeps giving the user chances in option if they entered an existing data
    IDstr = ""
    while lock:
        IDstr = input("Enter the ID of the New Student: ")
        check, IDstr = CheckId(IDstr)
        if check:
            # check = checkAlreadyExists(IDstr, studentList)
            check = True
            for i in studentList:
                if i.getId() == int(IDstr):
                    check = False
            if check:
                lock = False
            else:
                print("That ID already exist!")
                choice2 = input("Enter 1 if you want to try again: ")
```

```
            if choice2[0] != '1':
                print("No new record will be added!")
                lock = False
    else:
        print("No new record will be added!")
        lock = False
  return IDstr, check



#----------------------------------- Function: Check Semester ---------------------------------------------
def checkSemesterInfo(student):        # This function is to add a new semester for a student
  lock = True                          # And checks if this student already has this semester
  check = False
  semester="-"
  while lock:
    year = input("Enter the year in form of(118, 119, 120...): ")
    sem = input("Enter semester (1: for fall, 2: for spring, 3: for summer): ")
    if ( (sem[0]).isdigit() and len(year)>=3 ):
        seme = int(sem[0])
        if seme >= 1 and seme <=3 and (year[0:3]).isdecimal():
            check, semester = student.checkNewSemester(int(year[0:3]), int(sem[0]))
            lock = False
            if not check:
                print("This student already has this semester!")
        else:
            print("The data you entered is wrong!")
            ch = input("Enter 1 if you want to try again: ")
            if ch[0]!='1':
                print("No new semester will be added!")
                lock = False
    else:
        print("The data you entered is wrong!")
        ch = input("Enter 1 if you want to try again: ")
```

```
        if ch[0] != '1':
            print("No new semester will be added!")
            lock = False
    return semester, check



#----------------------------- Function: Find Student and check ID ----------------------------------------
def findIndexAndCheck():
    lock = True            # This function checks if the student exists
    check = False          # and gives his index in the student list
    IDstr = ""
    index = len(studentList)
    while lock:
        IDstr = input("Enter the ID of the Student: ")
        check, IDstr = CheckId(IDstr)
        if check:
            ID = int(IDstr)
            check = False
            for stu in studentList:
                if stu.getId() == ID:
                    check = True
                    index = studentList.index(stu)
            if check:
                lock = False
            else:
                print("That student doesn't exist!")
                choice2 = input("Enter 1 if you want to try again: ")
                if choice2[0] != '1':
                    print("This operation won't work!")
                    lock = False
        else:
            print("This operation won't work!")
            lock = False
```

```
    return index,check



#-------------------- Function: Check if course Exists ---------------------------------------------
def checkAndFindCourse():
    index= len(courseList)         # This function checks if the course exists
    lock = True                    # and returns its index in the courses list
    check = False
    while lock:
        name = input("Enter Course ID: ")
        name = name.upper()
        for course in courseList:
            if course.getName() == name:
                index = courseList.index(course)
                check = True
        if check:
            lock = False
        else:
            print("This course ID isn't available!")
            cho = input("Enter 1 if you want to try again: ")
            if cho[0] !='1':
                print("No grade will be changed")
                lock = False
    return index, check



#------------- Function: Replace Grade in File ------------------------------------------------
def replaceinFile(student, indexSem, indexCou, newGrade):
    file1 = open(student.getFile(), "r")                # This function searches for a grade
    lines = file1.readlines()                           # for a student and in his file
    file1.close()                                       # then changes it
    column= (13*(indexCou+1)) + 10
    lineList = list(lines[indexSem + 1])
```

```
    lineList[column] = newGrade[0]
    lineList[column+1] = newGrade[1]
    lines[indexSem + 1] = "".join(lineList)
    file2 = open(student.getFile(), "w")
    for line in lines:
        file2.write(line)


#----------------------- Function: Sort Semester List -------------------------------------------
def semesterSort():
    for i in range(0, len(semesterList)):              # This function sorts
        for j in range(i+1, len(semesterList)):        # the semester list by oldest to newest
            i0 = int(semesterList[i][0:4])
            j0 = int(semesterList[j][0:4])
            iS = int(semesterList[i][10])
            jS = int(semesterList[j][10])
            if ( i0 > j0 )  or ( i0==j0 and iS>jS):
                s = semesterList[i]
                semesterList[i] = semesterList[j]
                semesterList[j] = s


#================================= The Main ====================================================
if coursesExist:
    print("\n     Welcome to Student Record System\n"
        "*******************************************\n")
    user = input("Are you an Admin or a Student: ")     # The letters can be either capital
    user = user.lower()                                 # or small
    if user == "admin":
        again = True
        while again:
            print("\n     THE MENU  \n"                # the first index is considered
                  "******************\n"               # as the menu's choice
```

```
            "1- Add a New Record\n"
            "2- Add a New Semester for Student\n"
            "3- Update Grade\n"
            "4- Get Student Statistics\n"
            "5- Get Global Statistics\n"
            "6- Search Based on Credits or Grades\n"
            "others - Exit\n"
            "-----------------------------------")
        choice = "" # This loop is to make sure that the user enters at least something
        while len(choice) == 0:
            choice = input("Enter your choice here: ")
        # ---------------------------------------------------------------------------------
        if choice[0] == '1':       # To add a new student
            IDstr, check = EnterIDAndCheckIFItAlreadyExists()   # Enter id and check if it is
            if check:                                   # unique
                ID = int(IDstr)
                newStud = Student(ID, False)                # false as new and has no file
                stuFile = open("students.txt", "a")
                stuFile.write("\n" + IDstr)                 # add student to students file
                stuFile.close()
                studentList.append(newStud)                 # add student to students list
        # ---------------------------------------------------------------------------------
        elif choice[0] == '2':
            index, check = findIndexAndCheck()  # get student index and check if he exists
            if check:
                student = studentList[index]
                semester, check = checkSemesterInfo(student)    # check if this student
                if check:                               # already has this semester
                    student.semesters.append(semester) # add this new semester to student semesters
                    student.subjPerSem.append([])     # add a new list of courses for this semester
                    info = semester + " ;"            # info is the string to be added to file
                    courseName = ""
                    courseName = input("Enter a new course name or type exit to leave: ")
```

```
                courseName = courseName.upper()
                counter = 0     # Now a loop will go on as the user enters courses
                while courseName != "EXIT":     # The loop will end when the user types exit
                    lock = True
                    C = 0
                    while lock:
                        gradeStr = input("Enter the Grade: ")# the grade should be a decimal
                        if gradeStr[0:2].isdecimal() and len(gradeStr) >= 2:
                            grade = int(gradeStr[0:2])
                            if grade < 55:
                                grade = 55
                            check, C = student.addGrade(courseName, semester, grade)
                            lock = False
                        else:   # case of the user didn't enter a number
                            print("That's not a valid grade!")
                    if check:
                        if counter != 0:
                            info = info + ","   # add the course grade to the string for the file
                        info = info + " " + C.getName() + " " + str(C.getGrade())
                        counter += 1
                    else:   # if the user entered something unvalid
                        print("The course isn't available!")
                    courseName = input("Enter a new course name or type exit to leave: ")
                    courseName = courseName.upper() # course name should be in uppercase
                f = open(student.getFile(), "a")    # write all the info in the file
                f.write("\n" + info)
                f.close()
        # ------------------------------------------------------------------------------------
        elif choice[0] == '3':
            index, check = findIndexAndCheck()  # check if the student exists
            if check:
                student = studentList[index]
                index2, check = checkAndFindCourse()
```

```
            if check:
                course = student.allCourses[index2]  # gets and checks the course for the student
                if course.getGrade() != 0:  # check that the student took this course
                    newGradeStr = input("Enter new Grade: ")
                    if newGradeStr[0:2].isdecimal():    # check that the user entered a number
                        newGrade = int(newGradeStr)
                        if newGrade < 55:   # grades should be 55 or more
                            newGrade = 55
                        # Check for the semester to be edited in it's line in the file
                        semIndex, courInSemIndex = student.searchForSemesterOfCourse(course)
                        replaceinFile(student, semIndex, courInSemIndex, newGradeStr[0:2])
                    else:   # case grade wasn't right
                        print("That's not a valid grade!")
                else:       # case the course grade =0, which means the student didn.t take it
                    print("This student didn't take this course!")
        # ------------------------------------------------------------------------------------
        elif choice[0] == '4':
            index, check = findIndexAndCheck()  #checks for the student
            if check:
                student = studentList[index]
                print("Taken credits: ", student.getCredTaken())    # prints taken credit
                print("Passed credits: ", student.getCredPass(), "\n\n")# prints passed credit
                student.remainingCourses()  #Prints student's all remaining courses
                student.avgPerSem()     # Prints student's average per all semesters
                print("\nOverall Average: ", student.getAvg())  # Prints student's overall average
            else:
                print("Not valid")
        # ------------------------------------------------------------------------------------
        elif choice[0] == '5':
            print("\nOverall students average :", end=" ")# Prints the overall average
            cred = 0                                # For all students
            gra = 0
            for student in studentList:     # loop on all available students
```

```
            gra += (student.getAvg() * float(student.getCredTaken()))
            cred += student.getCredTaken()
        if (cred == 0):
            print("0")
        else:
            avg = float(gra) / float(cred)
            print(avg)
        if len(semesterList) > 0:
            semesterSort()
            print("\nSemester, overall average, average credit")
            for sem in semesterList:    # a loop to print the overall average
                stri = sem + ": "       # for all students in each semester
                avgs = 0
                cre = 0
                studPerSem = 0
                for s in studentList: # A loop for all students how have this semester
                    check, a, c = s.avgAndCredForSem(sem)
                    if check:
                        avgs += (a * c)
                        cre += c
                        studPerSem += 1
                overAllAvg = 0
                avgCred = 0
                if cre != 0:    # printing the overall average credits per semester
                    overAllAvg = float(avgs) / float(cre)
                    avgCred = float(cre) / float(studPerSem)
                stri += str(overAllAvg) + ", " + str(avgCred)
                print(stri)
        else:
            print("\nNothing Is available!")
        print("\nThe histogram of the averages should be displayed!")
        avGList = []        # Ploting the averages histogram
        for st in studentList:
```

```
            avGList.append(st.getAvg())
        plt.hist(avGList, bins=[50, 60, 70, 80, 90, 100], rwidth=0.95)
        plt.show()
    # ---------------------------------------------------------------------------------------
    elif choice[0] == '6':  # this choice to print all students who satisfy given criteria
        choice2 = ""
        while len(choice2) == 0:
            choice2 = input("enter 1 to search by credits, 2 to search by average: ")
        if choice2[0] == '1':
            cred = input("Enter Number of Credits to search for: ")
            if cred[0:2].isdecimal():
                credit = int(cred[0:2])
                choice3 = ""
                while len(choice3) == 0:
                    choice3 = input("Enter 1 to searched by more, 2 to search by less than: ")
                if (choice3[0] == '1' or choice3[0] == '2') and credit >= 0 and credit <= 69:
                    print("Student ID, Number of Passed Credits")
                    for student in studentList:
                        if choice3[0] == '1' and student.getCredPass() >= credit:
                            print(student.getId(), "   , ", student.getCredPass())
                        if choice3[0] == '2' and student.getCredPass() < credit:
                            print(student.getId(), "   , ", student.getCredPass())
                else:
                    print("That operation isn't avilable!")
            else:
                print("that's not a valid number of credits!")
        elif choice2[0] == '2':
            av = input("Enter the average to search for: ")
            avs = av.replace(".", "")
            if avs.isdecimal() and av.count('.') < 2:
                average = float(av)
                choice3 = ""
                while len(choice3) == 0:
```

```
                            choice3 = input("Enter 1 to searched by more, 2 to search by less than: ")
                        if (choice3[0] == '1' or choice3[0] == '2') and average >= 55 and average <= 99:
                            print("Student ID, Average")
                            for student in studentList:
                                if choice3[0] == '1' and student.getAvg() >= average:
                                    print(student.getId(), "   , ", student.getAvg())
                                if choice3[0] == '2' and student.getAvg() < average:
                                    print(student.getId(), "   , ", student.getAvg())
                        else:
                            if choice3[0] != '1' and choice3[0] != '2':
                                print("That operation isn't avilable!")
                            else:
                                print("that's not a valid average (range issue)!")
                    else:
                        print("that's not a valid average!")
                else:
                    print("you entered an invalid data!")
            # -------------------------------------------------------------------------------------
            else:
                again = False
                print("Program is Exiting..........")
    #^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    elif user == "student":
        student = 0     # Here it must be proven that you are a student
        index, again = findIndexAndCheck()  # checking if the student exisists
        if again:
            student = studentList[index]
        while again:
            print("\n     THE MENU  \n"
                  "******************\n"
                  "1- Get Student Statistics\n"
                  "2- Get Global Statistics\n"
                  "others - Exit\n"
```

```
            "-----------------------------------")
        choice = ""
        while len(choice) == 0:     # a loop until the user enters something
            choice = input("Enter your choice here: ")
        # ------------------------------------------------------------------------------------
        if choice[0] == '1':
            print("Taken credits: ", student.getCredTaken())  # prints taken credit
            print("Passed credits: ", student.getCredPass(), "\n\n")  # prints passed credit
            student.remainingCourses()  # Prints student's all remaining courses
            student.avgPerSem()  # Prints student's average per all semesters
            print("\nOverall Average: ", student.getAvg())  # Prints student's overall average
        # ------------------------------------------------------------------------------------
        elif choice[0] == '2':
            print("\nOverall students average :", end=" ")  # Prints the overall average
            cred = 0  # For all students
            gra = 0
            for student in studentList:  # loop on all available students
                gra += (student.getAvg() * float(student.getCredTaken()))
                cred += student.getCredTaken()
            if (cred == 0):
                print("0")
            else:
                avg = float(gra) / float(cred)
                print(avg)
            if len(semesterList) > 0:
                semesterSort()
                print("\nSemester, overall average, average credit")
                for sem in semesterList:  # a loop to print the overall average
                    stri = sem + ": "  # for all students in each semester
                    avgs = 0
                    cre = 0
                    studPerSem = 0
                    for s in studentList:  # A loop for all students how have this semester
```

```
                            check, a, c = s.avgAndCredForSem(sem)
                            if check:
                                avgs += (a * c)
                                cre += c
                                studPerSem += 1
                        overAllAvg = 0
                        avgCred = 0
                        if cre != 0:  # printing the overall average credits per semester
                            overAllAvg = float(avgs) / float(cre)
                            avgCred = float(cre) / float(studPerSem)
                        stri += str(overAllAvg) + ", " + str(avgCred)
                        print(stri)
                else:
                    print("\nNothing Is available!")
                print("\nThe histogram of the averages should be displayed!")
                avGList = []  # Ploting the averages histogram
                for st in studentList:
                    avGList.append(st.getAvg())
                plt.hist(avGList, bins=[50, 60, 70, 80, 90, 100], rwidth=0.95)
                plt.show()
            # ---------------------------------------------------------------------------------------
            else:
                again = False
                print("\nProgram is Exiting..........")
    # ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    else:
        print("\nThat's not a valid option, exiting.........")
```