

2023 Semester 1 - Main Final Exam

Coversheet

ACADEMIC INTEGRITY

Online exams are monitored for potential academic integrity issues, and breaches will be investigated. Review the [Student Charter](#) and your [Academic integrity responsibilities](#) and the [Integrity contract](#)

You are responsible for adhering to the exam conditions:

You cannot communicate or share any exam-related materials with another student or any other party during the exam. This includes third party communication or collaboration apps or websites.

Failure to comply with the examination conditions will be considered an academic integrity breach and will lead to proceedings under the [Academic Honesty in Coursework Policy 2015](#) or where applicable, the [University of Sydney \(Student Discipline\) Rule 2016](#). If you are found to engage in cheating, there may be serious consequences, including but not limited to failing the exam (and other penalties).

REFERENCE MATERIAL PERMITTED:

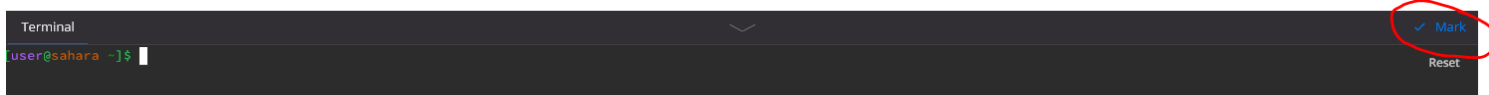
- Any course materials (textbook, lecture notes, tutorial sheets, weekly task sets, assignments, Ed)
- Ed workspace and Ed terminal
- Python 3 language reference (<https://docs.python.org/3>)

No other material is allowed, including Google, Stackoverflow, and Chegg. No collaborations are allowed, even if it is with your pet fish.

COMPUTER EXAM WRITING TIME: 72 hours

SUBMISSION:

You must click the **Mark** button to record each of your answers.



You may submit as many times as you wish within the time limit. Only the final submission will be marked.

The exam will finish on 10th June 2023 at 1700 hrs AEST (Sydney Local Time). You will not be able to make submissions after this time or view your exam after this time.

Note:

- The computer examination is not an assignment.
- Ed will enter moderation mode during the computer exam. You will still be able to post questions on Ed, but any post will need approval from a TA in charge.
- Staff may not be able to assist in your final exam
- You should write your assumption when you are asking for clarification of a question
- All final exam materials are strictly confidential. Do not record or share any materials from this exam. Penalties apply.
- This exam must be taken on a computer or laptop with satisfactory internet connectivity. It must NOT be taken on a mobile device.

Instruction for Question 1 and Question 2

Questions 1 and 2 of this exam are related to each other. In Question 1 you are writing a program based on the provided description of the question in `program.py`. In Question 2 you are designing test cases to test your program in Question 1. Please have this design in mind when you are writing your program in Question 1.

Design your test cases in Question 2 to verify all the functions that you wrote for different menu options of Question 1. You need to make sure these functions are implemented correctly. Write your testing plan first into the file `test_plan.md`. You are expected to explain your test cases and demonstrate how you will implement them. You must design at least 6 test cases for each function in Question 1: there must be 2 positive test cases, 2 negative test cases and 2 edge cases.

Then you need to write your test program for one of the functions of your code in `test.py` to verify your implementations for that function in Question 1. When you write the test case for this function, consider the test case should be its own function and you call your functions (test case) at the end.

Question 1 - Student Information Center [30 marks]

You are asked to create a program for a Student Information Center. This program gives these menu options to the users to choose from:

Main Menu

Choose any of these options:

1. Add a course
2. Add a student
3. Add a teacher
4. List all students
5. List all teachers
6. Search for a student by their name or student ID
7. List the teachers and their courses for a student
8. Show the GPA of a course

Enter zero to exit the program.

Enter your choice: #

This program saves its information inside three text files: `student_info.txt`, `course_info.txt` and `teacher_info.txt`. These three files might have existed before or they might have been new. If they exist your program will append to them, if they do not exist your program will create these files.

`student_info.txt` file

The student's records in this program are stored in a text file called `student_info.txt`. Each student has these details in this text file:

- Student ID
- Student name
- Program code
- Courses: including course code, course score

The format of `student_info.txt` file should be:

```
2067564342,Albert Smith,CS123,CSC300:65,ITS230:80
```



Notice the ':' between course code and course score.

`course_info.txt` file

The course information will be saved in a text file called `course_info.txt`, which has the following details:

- Course code
- Course name
- Credit hour

The format of `course_info.txt` file should be:

```
ITS230,Object Oriented Programming,3
CSC300,Introduction to programming,3
```

`teacher_info.txt` file

The teacher's details will be saved in a file called `teacher_info.txt` and has these details:

- Staff ID
- Staff name
- Courses codes

The format of `teacher_info.txt` file should be:

```
1089786,Franklin Dwyer,CSC230,ITS230
```



You have to strictly follow the file format in this description.

The program

The program continues to print the above menu until the user enters zero to exit. Before the program prompt the user with the menu, it adds an empty new line before the menu. Look at the sample run of the program at the end of the description.

Each menu option in your program has to be written as a function. You have to name these functions according to this naming convention:

1. Function for adding a course (function should be called: `add_course`)
2. Function for adding a student (function should be called: `add_student`)
3. Function for adding a teacher (function should be called: `add_teacher`)
4. Function for listing all students (function should be called: `list_students`)
5. Function for listing all teachers (function should be called: `list_teachers`)
6. Function to search for a student by their name or student ID (function should be called:

search_student)

7. Function for listing the teachers and their courses for a student (function should be called:

list_teachers_courses)

8. Function to show the GPA of a course (function should be called: **course_gpa)**

For example, in the menu option **1** , to add a product you are required to call **add_course()** .

The program continues to print the above menu until the user enters **0** to exit.



All of these functions do not take in any arguments or return any values EXCEPT for **student_gpa** which takes in a list as an argument, and returns a float value.



Your program is case-insensitive.



Please make sure your functions use the same names as specified above!

Adding a course (option 1)

The information for courses is saved in the **course_info.txt** . When the user chooses option 1 to add a course these details will be asked from them and will be saved in the **course_info.txt** :

```
Enter your choice: #1
Enter the course code: #CSC300
Enter the course name: #Introduction to programming
How many credit hours does this course have? #3
```

You are writing a function called **add_course** to run this option in the menu. That means when the user enters option 1 the function **add_course** will be called in your code. It asks the user to enter the course code, course name, and credit hour and then add this new course to the **course_info.txt** file. This function does not return any value.

If the course code exists in the **course_info.txt** file (course name can be repetitive), the user will get this error and the information about the course will not be saved in the file:

```
This course code exists in our database.
```

Sample Output:

```
Enter your choice: #1
Enter the course code: #CSC300
This course code exists in our database.
Enter the course code: #ITS230
Enter the course name: #Object Oriented Programming
How many credit hours does this course have? #3
```

The credit hour for each course should be an integer between 1 and 5 inclusive.

If the credit hour entered is not valid, the following error will be shown:

```
...
How many credit hours does this course have? #@
The credit hour for each course should be an integer between 1 and 5 inclusive.
How many credit hours does this course have? #1.5
The credit hour for each course should be an integer between 1 and 5 inclusive.
How many credit hours does this course have? #3
...
```

To check if the credit hour is valid, you must write a function `is_valid_credit_hour`, which should take one argument and return a boolean value indicating if the given credit hour is valid. This function will be later used in **Question 2 Part B** of the exam for testing.

```
is_valid_credit_hour('3') -> True
is_valid_credit_hour('1.5') -> False
is_valid_credit_hour('aa') -> False
```

Adding a student (option 2)

If the entered choice is `2` to add a student, the program will ask for all inputs related to the student which are student ID, name, the program code, and the number of courses for this student. For each course, the user will have to enter the course code and the score for that course. You are writing a function called `add_student` to run this option in the menu.

The details of the students will be stored in the `student_info.txt` and are linked to the details of the courses in `course_info.txt`. There should be an error handling placed to prevent incorrect inputs from the user (refer to the **Error Handling section** of the description).

This is the sample output of the program when the user enters `2`:

```
Enter your choice: #2
Enter the student ID: #2067564342
Enter the student name: #Albert Smith
Enter the program code: #CS123
How many courses does this student have? #2
Enter course code 1: #CSC300
Enter the student's score for this course: #65
Enter course code 2: #ITS230
Enter the student's score for this course: #80
```

If the user enters a course that doesn't exist in the `course_info.txt` file it will get an error:

```
Enter your choice: #2
Enter the student ID: #2067564342
Enter the student name: #Albert Smith
```

```
Enter the program code: #CS123
How many courses does this student have? #2
Enter course code 1: #CSC790
This course does not exist in our database.
Enter course code 1: #CSC300
Enter the student's score for this course: #65
Enter course code 2: #ITS230
Enter the student's score for this course: #80
```

If the student ID exists in the `student_info.txt` file (student name can be repetitive), the user will get this error and the information about the student will not be saved in the file:

```
This student ID exists in our database.
```

Sample Output:

```
Enter your choice: #2
Enter the student ID: #2067564342
This student ID exists in our database.
Enter the student ID: #2158432198
Enter the student name: #Eric Lim
Enter the program code: #CS123
How many courses does this student have? #2
Enter course code 1: #CSC300
Enter the student's score for this course: #67
Enter course code 2: #ITS230
Enter the student's score for this course: #90
```

The user can enter a maximum of 5 courses for each student each time, if they enter more than 5 courses they will get this error:

```
...
How many courses does this student have? #6
You can enter a number between 1 to 5.
How many courses does this student have? #3
...
```

If the number of courses entered is not an integer, the following error will be shown:

```
...
How many courses does this student have? 2.5
Number of courses should be an integer value.
How many courses does this student have? a
Number of courses should be an integer value.
How many courses does this student have? 2
Enter course code 1: csc300
...
```

Similarly for Scores:


```
...
```



```
Enter the student's score for this course: -2
The score for each course should be in the range of 0 to 100.
Enter the student's score for this course: a
Score should be a float value.
Enter the student's score for this course: 30.5
...
```

The user can enter each course code only once for each student. If they enter a duplicate course code they will get an error:

```
Enter your choice: #2
Enter the student ID: #2158432198
Enter the student name: #Eric Lim
Enter the program code: #CS123
How many courses does this student have? #2
Enter course code 1: #CSC230
Enter the student's score for this course: #65
Enter course code 2: #CSC230
This course has been entered before.
Enter course code 2: #CSC149
Enter the student's score for this course: #68
```

 These sample outputs show some of the errors that need to be handled, for the full list, please refer the **Error Handling section** down below in the description.

Adding a teacher (option 3)

If the user enters `3` to add a teacher the program would ask for all inputs related to the teacher which are staff ID, name, and the number of courses they are teaching. For each course, the user will have to enter the course code. You are writing a function called `add_teacher` to run this option in the menu.

The details will be stored in `teacher_info.txt`.

This is the sample output of the program when the user enters `3`:

```
Enter your choice: #3
Enter the staff ID: #1089786
Enter the teacher name: #Franklin Dwyer
How many courses does this teacher teach? #2
Enter course code 1: #CSC230
Enter course code 2: #ITS230
```

If the course does not exist, the user will get this error and the information about the course will not be saved:

```
Enter your choice: #3
Enter the staff ID: #1089786
Enter the teacher name: #Franklin Dwyer
How many courses does this teacher teach? #2
Enter course code 1: #CSC278
This course does not exist in our database.
Enter course code 1: #CSC230
```

If the staff ID exists in the `teacher_info.txt` file (teacher name can be repetitive), the user will get this error and the information about the teacher will not be saved in the file:

```
This teacher ID exists in our database.
```

Sample Output:

```
Enter your choice: #3
Enter the staff ID: #1045795
Enter the teacher name: #Simon Gray
How many courses does this teacher teach? #1
Enter course code 1: #CSC230
```

If the user enters a duplicate course code, they get this error:

```
Enter your choice: #3
Enter the staff ID: #1089786
Enter the teacher name: #Franklin Dwyer
How many courses does this teacher teach? #2
Enter course code 1: #CSC230
Enter course code 2: #CSC230
This course has been entered before.
Enter course code 2: #ITS230
```

The user can enter a maximum of 5 courses for each teacher each time, if they enter more than 5 courses they will get this error (They get the same error if they enter anything other than an integer between 1 to 5)"

```
...
How many courses does this teacher teach? #6
You can enter a number between 1 to 5.
How many courses does this teacher teach? #3
...
```

If the number of courses entered is not an integer, the following error will be shown:

```
...
How many courses does this teacher teach? 2.5
Number of courses should be an integer value.
```

```
How many courses does this teacher teach? a
Number of courses should be an integer value.
How many courses does this teacher teach? 1
Enter course code 1:
...
```



These sample outputs show some of the errors that need to be handled, for the full list, please refer the **Error Handling section** down below in the description.

Calculating GPA (Recursive function)

The process of calculating GPA will be done during the listing of all the students. You have to write a *recursive function* called **student_gpa** to calculate the GPA for students. Your recursive function should get a list of scores with the course code for each student from `student_info.txt` with their credit hours from the `course_info.txt`, and based on that calculate their GPA.

You can use this table to convert the scores for each course to "Grades" and then calculate the GPA based on the grades:

Scores obtained	Grade
[80 - 100]	4.00
[75 - 80)	3.67
[70 - 75)	3.33
[65 - 70)	3.00
[60 - 65)	2.67
[55 - 60)	2.33
[50 - 55)	2.00
[47 - 50)	1.67
[44 - 47)	1.33
[40 - 44)	1.00
Rest	0.00

The GPA is calculated based on the sum of grades multiplied by their credit hour, divided by the total credit hours for the student:

$$\text{GPA} = \frac{\text{Sum of grades multiplied by credit hours of each course}}{\text{Total credit hours}}$$

The function call for the recursive function will be a list of tuples, each tuple has a course code and a score for that course. The returning value from the recursive function is a float number, showing the GPA of the student. For example, if we were to use this function to calculate the GPA of a student with the two below courses, our function call would look something like this:

```
'CSC200' -> 3 Credit Hours
'ITS230' -> 3 Credit Hours

ls = [('CSC200',45), ('ITS230',90)]
student_gpa(ls) #2.665
```

This function can be used in the next menu option of the program when the user chooses to list all students. You may also find it useful in other options too.



When writing your recursive function identify the base case and recursive case and comment them in your code. Identifying base case and recursive cases has a portion of your mark for this question.



You might think of other non-recursive solutions which are easier to solve this task, but your program has to do this part of your code (calculating GPA) recursively, otherwise, no marks will be given to this section of your code.

List all students (option 4)

If the user inputs **4** in the original menu, they are getting a list of all students' details with their calculated GPA rounded to **2** decimal places (it would help to use the above recursive function for calculating GPA). You are writing a function called **list_students** to run this option in the menu.

Here is how the output of the program looks like when the user enters **4**:

(assume these students exist in `student_info.txt`)

```
Enter your choice: #4
Student ID: 2089786756
Student Name: Stanely Parade
Program Code: CS456
Course      Credit Hour      Score
ITS780      3              85.0
ITS569      3              87.0
ITS452      2              85.0
GPA is 4.00
```

```
Student ID: 2067564356
Student Name: Alex Frenandex
Program Code: CS349
Course      Credit Hour      Score
```

CSC300	3	75.0
ITS569	3	91.0
CSC560	2	65.0
CSC650	2	74.0

GPA is 3.57



You need to handle all errors in user inputs here as well, please refer the **Error Handling section** down below in the description

List all teachers (option 5)

You are writing a function called **list_teachers** to run this option in the menu.

When the user chooses this option, it will get the following output from the program:

(assume these teachers exist in `teacher_info.txt`)

```
Enter your choice: #5
Staff ID: 1089786
Teacher name: Franklin Dwyer
Teaches the following courses:
CSC300: Introduction to Programming
CSC340: Object Oriented Programming

Staff ID: 1076342
Teacher name: Dowryn Moore
Teaches the following courses:
CSC230: Statistics and Machine learning
ITS657: Introduction to Math
ITS231: Basics of Calculus

....
```



You need to handle all errors in user inputs here as well, please refer the **Error Handling section** down below in the description

Search for a student (option 6)

This is the output of the program when the user selects to search for a student in the main menu. Search can be done using the student ID or name of the student. You are writing a function called **search_student** to run this option in the menu.

Once the student ID or name has been entered, the student information will appear.



Note: GPA is rounded to **2** decimal places here as well.

(assume this student exists in `students_info.txt`)

```
Enter your choice: #6
Search students based on:
Select 1 for ID
Select 2 for Name
Enter your choice: #1
Enter student ID: #2067564356
Student ID: 2067564356
Student Name: Alex Frenandez
Program Code: CS349
Course      Credit Hour    Score
CSC480      3                75.0
ITS569      3                91.0
CSC560      2                65.0
CSC650      2                74.0
GPA is 3.57
```

If the user enters a student ID or name that does not exist in the system, it will get this error:

```
This student ID/name does not exist in our database.
```

Sample Output:

```
Enter your choice: #6
Search students based on:
Select 1 for ID
Select 2 for Name
Enter your choice: #2
Enter student name: #Matt Roberts
This student ID/name does not exist in our database.
Enter student name: #Eric Lim
Student ID: 2067123356
Student Name: Eric Lim
Program Code: CS123
Course      Credit Hour    Score
CSC230      3                45.0
ITS230      2                90.0
GPA is 2.67
```


If the user enters a student name that is duplicated in the `students_info.txt` file (Remember the student name can be duplicated, but the student ID has to be unique) the program will print all the records of duplicated student names from the file with their unique Student IDs.

For this example, imagine we have 2 students called "Eric Lim" in our database:

```
Enter your choice: #6
Search students based on:
Select 1 for ID
Select 2 for Name
Enter your choice: #2
```

```
Enter student name: #Eric Lim
Student ID: 2067123356
Student Name: Eric Lim
Program Code: CS123
Course      Credit Hour    Score
CSC230      3              45.0
ITS230      2              90.0
GPA is 2.67
```

```
Student ID: 2054318123
Student Name: Eric Lim
Program Code: CS123
Course      Credit Hour    Score
CSC480      3              75.0
ITS569      3              91.0
CSC560      2              65.0
CSC650      2              74.0
GPA is 3.57
```

 You need to handle all errors in user inputs here as well, please refer the **Error Handling section** down below in the description

List the teachers and their courses for a student (option 7)

When the user chooses this option, the program has to look through the three available files to find the relevant information for the user. The user enters a student ID and will get the name of the teachers that teach the courses they are enrolled in.

You are writing a function called `list_teachers_courses` to run this option in the menu.

```
Enter your choice: #7
Enter student ID: #2063643557
Teacher Name: Franklin Dwyer
Course Name: Introduction to programming

Teacher Name: Deborah Roger
Course Name: Statistics and Machine learning
```

If the student is enrolled in a course `Statistics and Machine learning` and this course is taught by multiple teachers, all of them need to be printed:

```
Enter your choice: 7
Enter student ID: 2032154400
Teacher Name: Dowryn Moore
Course Name: Statistics and Machine learning

Teacher Name: Hailey Franklin
Course Name: Statistics and Machine learning
```

If there is a course with no associated teacher, they will only get the name of the course and a message like this:

```
....  
Course Name: Basics of Calculus  
The teacher for this course is not specified.
```

If the user enters a student ID that does not exist in the system, it will get this error:

```
This student ID does not exist in our database.
```

Sample Output:

```
Enter your choice: #7  
Enter student ID: #2164532999  
This student ID does not exist in our database.  
Enter student ID: 2063643557  
Teacher Name: Franklin Dowyer  
Course Name: Introduction to programming  
  
Teacher Name: Deborah Roger  
Course Name: Statistics and Machine learning
```



These sample outputs show some of the errors that need to be handled, for the full list, please refer the **Error Handling section** down below in the description

Show the GPA of a course (option 8)

When the user chooses this option the program should produce the GPA for a course searched by the course ID. You are writing a function called **course_gpa** to run this option in the menu.

The output of the program is similar to this:

```
Enter your choice: #8  
Enter the course code: #CSC300  
GPA for this course is 2.67
```

The GPA is calculated based on the above-mentioned formula for GPA and is considering the scores of all the students that enrolled in this course so far.

If the user enters a course code that does not exist in the system, it will get this error:

```
The course code does not exist in our database.
```

Sample output:

```
Enter your choice: #8
```



```
Enter the course code: #MAT998
The course code does not exist in our database.
Enter the course code: CSC300
GPA for this course is 2.67
```

i Note: GPA is rounded to **2** decimal places here as well.

i You need to handle all errors in user inputs here as well, please refer the **Error Handling section** down below in the description

Exiting the program

When the user enters zero as an option the program should exit with this message:

```
Enter your choice: #0
Exiting the program...
```

Error handling

All inputs in your program should be checked:

- Wrong input for menu selection, anything other than the specified menu options is not accepted. The error message is: `Wrong menu selection.`
- The length of the student ID should be 10 characters. The error message is: `The length of the student ID should be 10 characters.`
- The length of the teacher ID should be 7 characters. The error message is: `The length of the staff ID should be 7 characters.`
- The length of the program code entered should be 5. The error message is: `The length of the program code entered should be 5.`
- The length of course codes entered should be 6. The error message is: `The length of course codes entered should be 6.`
- The length of the student, teacher, and course name should be between 1 and 40 characters inclusive. The error message is: `The length of the name should be between 1 and 40 characters inclusive.`
- The student, teacher, and course names should be alphabetic. The error message is: `The name should be alphabetic.`
- The credit hour for each course should be an integer number more than 0 and less or equal to 5. The error message is: `The credit hour for each course should be an integer between 1 and 5 inclusive.`
- The score for each course should be in the range of 0 to 100. The error message is: `The score for each course should be in the range of 0 to 100.`

- The score for each course should be a float value. The error message is: `Score should be a float value.`
- Your program is case-insensitive, and if the user enters anything in capital letters or lower letters it will be treated the same.



You can assume that errors not mentioned in the description will not be tested.



Errors are ordered by priority e.g Length of name would be checked before determining whether it is alphabetic or not

A Sample Program Execution

Main Menu

Choose any of these options:

1. Add a course
2. Add a student
3. Add a teacher
4. List all students
5. List all teachers
6. Search for a student by their name or student ID
7. List the teachers and their courses for a student
8. Show the GPA of a course

Enter zero to exit the program.

Enter your choice: 1

Enter the course code: CSC300

Enter the course name: Introduction to Programming

How many credit hours does this course have? 3

Main Menu

Choose any of these options:

1. Add a course
2. Add a student
3. Add a teacher
4. List all students
5. List all teachers
6. Search for a student by their name or student ID
7. List the teachers and their courses for a student
8. Show the GPA of a course

Enter zero to exit the program.

Enter your choice: 2

Enter the student ID: 2067564342

Enter the student name: Albert Smith

Enter the program code: CS123

How many courses does this student have? 1

Enter course code 1: CSC790

This course does not exist in our database.

Enter course code 1: csc300

Enter the student's score for this course: 65.5

Main Menu

Choose any of these options:

1. Add a course
2. Add a student
3. Add a teacher
4. List all students
5. List all teachers
6. Search for a student by their name or student ID
7. List the teachers and their courses for a student
8. Show the GPA of a course

Enter zero to exit the program.

Enter your choice: 3

Enter the staff ID: 1089786

Enter the teacher name: Franklin Dwyer

How many courses does this teacher teach? 1

Enter course code 1: CsC300

Main Menu

Choose any of these options:

1. Add a course
2. Add a student
3. Add a teacher
4. List all students
5. List all teachers
6. Search for a student by their name or student ID
7. List the teachers and their courses for a student
8. Show the GPA of a course

Enter zero to exit the program.

Enter your choice: 4

Student ID: 2067564342

Student Name: Albert Smith

Program Code: CS123

Course	Credit Hour	Score
csc300	3	65.5

GPA is 3.00

Main Menu

Choose any of these options:

1. Add a course
2. Add a student
3. Add a teacher
4. List all students
5. List all teachers
6. Search for a student by their name or student ID
7. List the teachers and their courses for a student
8. Show the GPA of a course

Enter zero to exit the program.

Enter your choice: 5

Staff ID: 1089786

Teacher name: Franklin Dwyer

Teaches the following courses:

CsC300: Introduction to Programming

Main Menu

Choose any of these options:

1. Add a course
2. Add a student
3. Add a teacher
4. List all students
5. List all teachers
6. Search for a student by their name or student ID
7. List the teachers and their courses for a student
8. Show the GPA of a course

Enter zero to exit the program.

Enter your choice: a

Wrong menu selection

Main Menu

Choose any of these options:

1. Add a course
2. Add a student
3. Add a teacher
4. List all students
5. List all teachers
6. Search for a student by their name or student ID
7. List the teachers and their courses for a student
8. Show the GPA of a course

Enter zero to exit the program.

Enter your choice: 0

Exiting the program...

Testing your Code

This program does not have thorough public test cases to check all the functionalities of your code. You are responsible for testing your code thoroughly. Question 2 in this exam is about designing your test cases for this program and also implementing one of your test cases for this program.



Hardcoding to pass any of the testcases is forbidden in your code and will not receive any marks.



You cannot use the **global** keyword in your code.

Question 2 - Writing test cases for Question 1 [20 marks]

Consider the programming problem description of Question 1 - Student Information Center. You will need to write tests for the following functions of your program:

1. Function for adding a course (function: `add_course`)
2. Function for adding a student (function: `add_student`)
3. Function for adding a teacher (function: `add_teacher`)
4. Function for listing all students (function: `list_students`)
5. Function for listing all teachers (function: `list_teachers`)
6. Function to search for a student by their name or student ID (function: `search_student`)
7. Function for listing the teachers and their courses for a student (function: `list_teachers_courses`)
8. Function to show the GPA of a course (function: `course_gpa`)
9. Additionally, the recursive function to calculate the GPA for students (function: `student_gpa`)



Just to make it clear, you will answer part A of this question in `test_plan.md` and part B of this question in `test.py` .

Part A [10 marks]

You can start by describing the steps to test a program solution based on the question description. You will plan your test cases in `test_plan.md` . You are expected to explain your test cases and demonstrate how you will implement them.

You are testing all the functionality of functions for each menu option of Question 1. You must have at least 6 test cases for each function: 2 positive test cases, 2 negative test cases, and 2 edge cases. Your answer must include the test objectives and example test cases (2 positive cases, 2 negative cases, and 2 edge cases for each function). The test case descriptions must include details of the inputs and expected output. The file `test_plan.md` will be written in the Question 1 workspace.

Note that there might be a chance that you can not add any negative or edge cases for some functions, in those cases simply exclude them. As an example, if a function can only be tested with positive cases, you'll only have to write 2 positive test cases. However, it needs to be justified why these cases cannot be done.

Note that this part does not require any code, and ideally should not.

Part B [10 marks]

There is one function for which you must implement the test cases, requiring code:

`is_valid_credit_hour` in the file `test.py`. The test program will be run against various implementations of the functions: some correct implementations and many incorrect implementations. This is to ensure your test program can catch a wide range of bugs. You are **not** required to implement the function in this part, just to write tests in `test.py` (you implemented this function in Question 1).

The test cases for this function should consider all functionalities of your code, and consider the 6 positive, negative, and edge cases from Part A. You must implement at least 2 positive test cases, 2 negative test cases, and 2 edge cases. The file `test.py` will be implemented in the Question 1 workspace.

The test program must print **one** line to the terminal in the following format: `<function_name> has <pass_status>`. If the function being tested failed one or more of your tests, the test suite should print `<function_name> has failed. once only`. If the function being tested passes all tests, the test suite should print `<function_name> has passed. once only`.



The correct implementation must be identified by your test suite in order to receive marks for identifying the incorrect implementations. The marking script first calls your test suite on the correct version and then an incorrect version.

Example usages of `test.py`:

```
$ python test.py
is_valid_credit_hour has passed.
$
```

```
$ python test.py
is_valid_credit_hour has failed.
$
```



Note: You will **not** be able to run the file, but you can check your implementation using the test cases

Function specifications



Ensure you are testing the function, not implementing it

```
is_valid_credit_hour(credit_hour: str) -> bool
```

`is_valid_credit_hour()` accepts one positional argument `credit_hour` and returns a boolean. The function returns `True` if the credit hour is valid and `False` if the credit hour is invalid.

Validation Rule:

- The credit hour for each course should be a string representation of an integer between 1 and 5 inclusive.

Question 3 - Extension of Past Game - Mousehunt [50 marks]

You are given additional features to add to the Mousehunt game from the in-semester coursework of Assignment 1. Your task is to create a module called `fe` to implement the given functions and then call these functions to produce the correct behaviour. The original gameplay must not be modified with the additional features.

In this exam, you may assume that The Cheese Shop only sells `cheddar` and players can only catch `Brown` mouse during The Hunt. Similar to Assignment 1, players will always enter valid inputs when prompted *during the game*. You may choose to use your previous work from Assignment 1 or use the provided solutions which are pre-loaded in your workspace.



If you are to use your previous work from Assignment 1, you **must** ensure that `game.py` has a `main` function that will run your entire program, otherwise you will not pass any of the **Auto Pass** test cases and the first two test cases for **Q1**.

Question 1 - Log Events (15 Marks)

Your goal in this task is to log the major events that happen in the game and save these events into a file labelled as the player's in-game name e.g. `<name>.txt`. These are the major events that can happen in the game and must be included in the log when it happens:

- **Start game** - when player starts the game. This is when the game menu launches.
- **Start shop** - when player enters The Cheese Shop
- **Bought `x` cheddar** - when players purchase `x` amount of cheddar cheese in the The Cheese Shop. This event can only happen in The Cheese Shop
- **End shop** - when player leaves The Cheese Shop
- **Start hunt** - when players start the Hunt feature
- **Caught a Brown mouse!** - when players caught a Brown mouse during the Hunt
- **Nothing happens.** - when players attempt to Hunt but fails to attract a Brown mouse while their trap is armed with cheddar.
- **Nothing happens. You are out of cheese!** - when players attempt to Hunt but fails to attract a Brown mouse cause their trap is not armed with cheddar.
- **Do nothing** - when players attempt to Hunt but fails to sound the horn correctly.
- **End hunt** - when players leave the Hunt feature
- **End game** - when players exit the game.

Write a function called `refresh_file` that accepts one argument representing the absolute path to a file. If the file exists, it discards all the contents in this file and creates an empty file at the given path

to prepare for the writing of the events. Else, the function displays the message `File path does not exist.` . You may assume that only text files will be passed into this function.

Write a function called `log_events` that has two `str` parameters: `event` and `filename` . The second parameter has a default value set to `/home/saved/temp.txt` . If no arguments are provided, the function by default will write the event to the default file. The function returns `True` if the event, represented by `event` is successfully written to the file called `filename` . Else, the function does nothing and returns `False` . New events will always be added to the bottom of the file followed by a new line. If the `filename` argument is a valid path but the file does not exist, a new file should be created and the event written at the start of the file.

Modify `game.py` program from Assignment 1 to log the major events into the file `/home/saved/temp.txt` when it happens. You must call the `log_events` to accomplish this. The input/output of the actual program from Assignment 1 should not change with this addition.

If the player's name is valid (fulfills the conditions in Assignment 1 Question 2) when players opt to `Exit game` , a text file with the player's name is created and the contents in `/home/saved/temp.txt` is copied over. This text file will always be saved in the directory `/home/saved/` . The file is refreshed each time the game is restarted. For example, player Tom plays the game twice, the file contents stored in `temp.txt` and subsequently `Tom.txt` will only be the events that happen in the latest game since the first attempt is overridden.

Program Execution

The auto-marker will only start `game.py` and `fe.py` to check the files produced by your program. These are examples text input/output and file outputs for four different players - `Tom` , `Jerry` , `Bob` and `COMP9001` .

- Player Tom skips training and purchases 5 cheddar for their hunt. They continue hunting until he is out of cheese before ending his hunt and stopping the game.

```
$ python3 game.py
Mousehunt

      ____()()
      /      @@
`~~~~~\_;m__m._>o

Inspired by Mousehunt© Hitgrab
Programmer - An INFO1110/COMP9001 Student
Mice art - Joan Stark

What's ye name, Hunter?
#Tom
Welcome to the Kingdom, Hunter Tom!
Before we begin, let's train you up!
Press "Enter" to start training or "skip" to Start Game: #skip
```

What do ye want to do now, Hunter Tom?

1. Exit game
2. Join the Hunt
3. The Cheese Shop

#3

Welcome to The Cheese Shop!

Cheddar - 10 gold

How can I help ye?

1. Buy cheese
2. View inventory
3. Leave shop

#1

You have 125 gold to spend.

State [cheese quantity]: #cheddar 5

Successfully purchase 5 cheddar.

How can I help ye?

1. Buy cheese
2. View inventory
3. Leave shop

#3

What do ye want to do now, Hunter Tom?

1. Exit game
2. Join the Hunt
3. The Cheese Shop

#2

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Caught a Brown mouse!

My gold: 200, My points: 115

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Caught a Brown mouse!

My gold: 325, My points: 230

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Nothing happens.

My gold: 325, My points: 230

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Nothing happens.

My gold: 325, My points: 230

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Nothing happens.

My gold: 325, My points: 230

Sound the horn to call for the mouse...

```

Sound the horn by typing "yes": #yes
Nothing happens. You are out of cheese!
My gold: 325, My points: 230

Sound the horn to call for the mouse...
Sound the horn by typing "yes": #yes
Nothing happens. You are out of cheese!
My gold: 325, My points: 230

Do you want to continue to hunt? yes
Sound the horn to call for the mouse...
Sound the horn by typing "yes": #stop hunt

What do ye want to do now, Hunter Tom?
1. Exit game
2. Join the Hunt
3. The Cheese Shop
#1

```

These are the contents stored in `Tom.txt` at the end of the game.

```

Start game
Start shop
Bought 5 cheddar
End shop
Start hunt
Caught a Brown mouse!
Caught a Brown mouse!
Nothing happens.
Nothing happens.
Nothing happens.
Nothing happens. You are out of cheese!
Nothing happens. You are out of cheese!
End hunt
End game

```

- Player Jerry half-heartedly does the training, purchases 3 cheddar and hunts. Upon finishing all their cheese during the hunt, they stop hunt and immediately purchases another 5 cheddar to continue their hunt. They call it a day after finishing all the cheese.

```

$ python3 game.py
Mousehunt

      ____()()
     /      @@
`~~~~~\_;m__m._>o

Inspired by Mousehunt© Hitgrab
Programmer - An INF01110/COMP9001 Student
Mice art - Joan Stark

What's ye name, Hunter?
#Jerry

```

Welcome to the Kingdom, Hunter Jerry!
Before we begin, let's train you up!
Press "Enter" to start training or "skip" to Start Game: #

Larry: I'm Larry. I'll be your hunting instructor.
Larry: Let's go to the Meadow to begin your training!
Press Enter to travel to the Meadow...#
Travelling to the Meadow...
Larry: This is your camp. Here you'll set up your mouse trap.
Larry: Let's get your first trap...
Press Enter to view traps that Larry is holding...#
Larry is holding...
Left: High Strain Steel Trap
Right: Hot Tub Trap
Select a trap by typing "left" or "right": #left
Larry: Excellent choice.
Your "High Strain Steel Trap" is now set!
Larry: You need cheese to attract a mouse.
Larry places one cheddar on the trap!
Sound the horn to call for the mouse...
Sound the horn by typing "yes": #
Nothing happens.
To catch a mouse, you need both trap and cheese!

Press Enter to continue training and "no" to stop training: #no

What do ye want to do now, Hunter Jerry?

1. Exit game
2. Join the Hunt
3. The Cheese Shop

#3

Welcome to The Cheese Shop!
Cheddar - 10 gold

How can I help ye?

1. Buy cheese
2. View inventory
3. Leave shop

#1

You have 125 gold to spend.
State [cheese quantity]: #cheddar 3
Successfully purchase 3 cheddar.

How can I help ye?

1. Buy cheese
2. View inventory
3. Leave shop

#3

What do ye want to do now, Hunter Jerry?

1. Exit game
2. Join the Hunt
3. The Cheese Shop

#2

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Nothing happens.

My gold: 95, My points: 0

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Caught a Brown mouse!

My gold: 220, My points: 115

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Nothing happens.

My gold: 220, My points: 115

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #stop hunt

What do ye want to do now, Hunter Jerry?

1. Exit game
2. Join the Hunt
3. The Cheese Shop

#3

Welcome to The Cheese Shop!

Cheddar - 10 gold

How can I help ye?

1. Buy cheese
2. View inventory
3. Leave shop

#1

You have 220 gold to spend.

State [cheese quantity]: #cheddar 5

Successfully purchase 5 cheddar.

How can I help ye?

1. Buy cheese
2. View inventory
3. Leave shop

#3

What do ye want to do now, Hunter Jerry?

1. Exit game
2. Join the Hunt
3. The Cheese Shop

#2

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Nothing happens.

My gold: 170, My points: 115

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Caught a Brown mouse!

My gold: 295, My points: 230

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Nothing happens.

My gold: 295, My points: 230

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Caught a Brown mouse!

My gold: 420, My points: 345

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Caught a Brown mouse!

My gold: 545, My points: 460

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Nothing happens. You are out of cheese!

My gold: 545, My points: 460

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #stop hunt

What do ye want to do now, Hunter Jerry?

1. Exit game
 2. Join the Hunt
 3. The Cheese Shop
- #1

These are the contents stored in `Jerry.txt` at the end of the game.

Start game

Start shop

Bought 3 cheddar

End shop

Start hunt

Nothing happens.

Caught a Brown mouse!

Nothing happens.

End hunt

Start shop

Bought 5 cheddar

End shop

Start hunt

Nothing happens.

Caught a Brown mouse!

Nothing happens.

Caught a Brown mouse!

Caught a Brown mouse!

Nothing happens. You are out of cheese!

End hunt

- Player Bob skips training, purchases 10 cheddar and hunts. But, they couldn't get the horn to work correctly twice before continuing to hunt until their cheddar runs out.

```
$ python3 game.py
```

```
Mousehunt
```

```

      ____()()
     /      @@
`~~~~~\_;m__m._>o

```

```
Inspired by Mousehunt© Hitgrab
```

```
Programmer - An INF01110/COMP9001 Student
```

```
Mice art - Joan Stark
```

```
What's ye name, Hunter?
```

```
#Bob
```

```
Welcome to the Kingdom, Hunter Bob!
```

```
Before we begin, let's train you up!
```

```
Press "Enter" to start training or "skip" to Start Game: #skip
```

```
What do ye want to do now, Hunter Bob?
```

```
1. Exit game
```

```
2. Join the Hunt
```

```
3. The Cheese Shop
```

```
#3
```

```
Welcome to The Cheese Shop!
```

```
Cheddar - 10 gold
```

```
How can I help ye?
```

```
1. Buy cheese
```

```
2. View inventory
```

```
3. Leave shop
```

```
#1
```

```
You have 125 gold to spend.
```

```
State [cheese quantity]: #cheddar 10
```

```
Successfully purchase 10 cheddar.
```

```
How can I help ye?
```

```
1. Buy cheese
```

```
2. View inventory
```

```
3. Leave shop
```

```
#3
```

```
What do ye want to do now, Hunter Bob?
```

```
1. Exit game
```

```
2. Join the Hunt
```

```
3. The Cheese Shop
```

```
#2
```

```
Sound the horn to call for the mouse...
```

```
Sound the horn by typing "yes": #yes
```

```
Nothing happens.
```

My gold: 25, My points: 0

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #no

Do nothing.

My gold: 25, My points: 0

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #no

Do nothing.

My gold: 25, My points: 0

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Caught a Brown mouse!

My gold: 150, My points: 115

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Nothing happens.

My gold: 150, My points: 115

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Nothing happens.

My gold: 150, My points: 115

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Nothing happens.

My gold: 150, My points: 115

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Nothing happens.

My gold: 150, My points: 115

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Nothing happens.

My gold: 150, My points: 115

Do you want to continue to hunt? #yes

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Caught a Brown mouse!

My gold: 275, My points: 230

Sound the horn to call for the mouse...

Sound the horn by typing "yes": #yes

Nothing happens.

My gold: 275, My points: 230

Sound the horn to call for the mouse...


```
Sound the horn by typing "yes": #yes
Nothing happens.
My gold: 275, My points: 230

Sound the horn to call for the mouse...
Sound the horn by typing "yes": #yes
Nothing happens. You are out of cheese!
My gold: 275, My points: 230

Sound the horn to call for the mouse...
Sound the horn by typing "yes": #yes
Nothing happens. You are out of cheese!
My gold: 275, My points: 230

Sound the horn to call for the mouse...
Sound the horn by typing "yes": #stop hunt

What do ye want to do now, Hunter Bob?

1. Exit game
2. Join the Hunt
3. The Cheese Shop
#1
```

These are the contents stored in `Bob.txt` at the end of the game.

```
Start game
Start shop
Bought 10 cheddar
End shop
Start hunt
Nothing happens.
Do nothing.
Do nothing.
Caught a Brown mouse!
Nothing happens.
Nothing happens.
Nothing happens.
Nothing happens.
Nothing happens.
Caught a Brown mouse!
Nothing happens.
Nothing happens.
Nothing happens. You are out of cheese!
Nothing happens. You are out of cheese!
End hunt
End game
```

- Player `COMP9001` purchases cheese in the sequence of `1` , `2` , `3` on their first shop visit until they are unable to buy cheese.

```
$ python3 game.py
Mousehunt
```

```
      ____()()
      /      @@
`~~~~~\_;m__m._>o
```

Inspired by Mousehunt© Hitgrab
Programmer - An INF01110/COMP9001 Student
Mice art - Joan Stark

What's ye name, Hunter?
COMP9001
Welcome to the Kingdom, Hunter COMP9001!
Before we begin, let's train you up!
Press "Enter" to start training or "skip" to Start Game: skip

What do ye want to do now, Hunter COMP9001?

1. Exit game
 2. Join the Hunt
 3. The Cheese Shop
- 3

Welcome to The Cheese Shop!
Cheddar - 10 gold

How can I help ye?

1. Buy cheese
 2. View inventory
 3. Leave shop
- 1

You have 125 gold to spend.
State [cheese quantity]: cheddar 1
Successfully purchase 1 cheddar.

How can I help ye?

1. Buy cheese
 2. View inventory
 3. Leave shop
- 1

You have 115 gold to spend.
State [cheese quantity]: cheddar 2
Successfully purchase 2 cheddar.

How can I help ye?

1. Buy cheese
 2. View inventory
 3. Leave shop
- 1

You have 95 gold to spend.
State [cheese quantity]: cheddar 3
Successfully purchase 3 cheddar.

How can I help ye?

1. Buy cheese
2. View inventory
3. Leave shop

1

You have 65 gold to spend.

State [cheese quantity]: cheddar 4

Successfully purchase 4 cheddar.

How can I help ye?

1. Buy cheese
2. View inventory
3. Leave shop

1

You have 25 gold to spend.

State [cheese quantity]: cheddar 5

Insufficient gold.

How can I help ye?

1. Buy cheese
2. View inventory
3. Leave shop

3

What do ye want to do now, Hunter COMP9001?

1. Exit game
2. Join the Hunt
3. The Cheese Shop

2

Sound the horn to call for the mouse...

Sound the horn by typing "yes": yes

Caught a Brown mouse!

My gold: 150, My points: 115

Sound the horn to call for the mouse...

Sound the horn by typing "yes": yes

Caught a Brown mouse!

My gold: 275, My points: 230

Sound the horn to call for the mouse...

Sound the horn by typing "yes": yes

Caught a Brown mouse!

My gold: 400, My points: 345

Sound the horn to call for the mouse...

Sound the horn by typing "yes": yes

Caught a Brown mouse!

My gold: 525, My points: 460

Sound the horn to call for the mouse...

Sound the horn by typing "yes": yes

Nothing happens.

My gold: 525, My points: 460

Sound the horn to call for the mouse...

Sound the horn by typing "yes": yes

Caught a Brown mouse!

My gold: 650, My points: 575

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Nothing happens.
My gold: 650, My points: 575

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Caught a Brown mouse!
My gold: 775, My points: 690

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Caught a Brown mouse!
My gold: 900, My points: 805

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Nothing happens.
My gold: 900, My points: 805

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Nothing happens. You are out of cheese!
My gold: 900, My points: 805

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Nothing happens. You are out of cheese!
My gold: 900, My points: 805

Sound the horn to call for the mouse...
Sound the horn by typing "yes": stop hunt

What do ye want to do now, Hunter COMP9001?

1. Exit game
 2. Join the Hunt
 3. The Cheese Shop
- 3

Welcome to The Cheese Shop!
Cheddar - 10 gold

How can I help ye?

1. Buy cheese
2. View inventory
3. Leave shop

1

You have 900 gold to spend.
State [cheese quantity]: cheddar 9
Successfully purchase 9 cheddar.

How can I help ye?

1. Buy cheese
2. View inventory

3. Leave shop

1

You have 810 gold to spend.

State [cheese quantity]: cheddar 10

Successfully purchase 10 cheddar.

How can I help ye?

1. Buy cheese

2. View inventory

3. Leave shop

1

You have 710 gold to spend.

State [cheese quantity]: cheddar 11

Successfully purchase 11 cheddar.

How can I help ye?

1. Buy cheese

2. View inventory

3. Leave shop

1

You have 600 gold to spend.

State [cheese quantity]: cheddar 12

Successfully purchase 12 cheddar.

How can I help ye?

1. Buy cheese

2. View inventory

3. Leave shop

1

You have 480 gold to spend.

State [cheese quantity]: cheddar 13

Successfully purchase 13 cheddar.

How can I help ye?

1. Buy cheese

2. View inventory

3. Leave shop

3

What do ye want to do now, Hunter COMP9001?

1. Exit game

2. Join the Hunt

3. The Cheese Shop

2

Sound the horn to call for the mouse...

Sound the horn by typing "yes": yes

Caught a Brown mouse!

My gold: 475, My points: 920

Sound the horn to call for the mouse...

Sound the horn by typing "yes": yes

Nothing happens.

My gold: 475, My points: 920

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Nothing happens.
My gold: 475, My points: 920

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Caught a Brown mouse!
My gold: 600, My points: 1035

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Caught a Brown mouse!
My gold: 725, My points: 1150

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Nothing happens.
My gold: 725, My points: 1150

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Caught a Brown mouse!
My gold: 850, My points: 1265

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Caught a Brown mouse!
My gold: 975, My points: 1380

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Nothing happens.
My gold: 975, My points: 1380

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Nothing happens.
My gold: 975, My points: 1380

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Nothing happens.
My gold: 975, My points: 1380

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Caught a Brown mouse!
My gold: 1100, My points: 1495

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Nothing happens.
My gold: 1100, My points: 1495

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Nothing happens.
My gold: 1100, My points: 1495

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Nothing happens.
My gold: 1100, My points: 1495

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Nothing happens.
My gold: 1100, My points: 1495

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Caught a Brown mouse!
My gold: 1225, My points: 1610

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Nothing happens.
My gold: 1225, My points: 1610

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Nothing happens.
My gold: 1225, My points: 1610

Sound the horn to call for the mouse...
Sound the horn by typing "yes": ye
Do nothing.
My gold: 1225, My points: 1610

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Nothing happens.
My gold: 1225, My points: 1610

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Nothing happens.
My gold: 1225, My points: 1610

Do you want to continue to hunt? yes
Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes
Caught a Brown mouse!
My gold: 1350, My points: 1725

Sound the horn to call for the mouse...
Sound the horn by typing "yes": yes

Caught a Brown mouse!

My gold: 1475, My points: 1840

Sound the horn to call for the mouse...

Sound the horn by typing "yes": yes

Caught a Brown mouse!

My gold: 1600, My points: 1955

Sound the horn to call for the mouse...

Sound the horn by typing "yes": yes

Caught a Brown mouse!

My gold: 1725, My points: 2070

Sound the horn to call for the mouse...

Sound the horn by typing "yes": yes

Caught a Brown mouse!

My gold: 1850, My points: 2185

Sound the horn to call for the mouse...

Sound the horn by typing "yes": yes

Nothing happens.

My gold: 1850, My points: 2185

Sound the horn to call for the mouse...

Sound the horn by typing "yes": yes

Nothing happens.

My gold: 1850, My points: 2185

Sound the horn to call for the mouse...

Sound the horn by typing "yes": yes

Caught a Brown mouse!

My gold: 1975, My points: 2300

Sound the horn to call for the mouse...

Sound the horn by typing "yes": yes

Caught a Brown mouse!

My gold: 2100, My points: 2415

Sound the horn to call for the mouse...

Sound the horn by typing "yes": stop hunt

What do ye want to do now, Hunter COMP9001?

1. Exit game

2. Join the Hunt

3. The Cheese Shop

3

Welcome to The Cheese Shop!

Cheddar - 10 gold

How can I help ye?

1. Buy cheese

2. View inventory

3. Leave shop

2

Gold - 2100
Cheddar - 25
Trap - Cardboard and Hook Trap

How can I help ye?

1. Buy cheese
 2. View inventory
 3. Leave shop
- 3

What do ye want to do now, Hunter COMP9001?

1. Exit game
 2. Join the Hunt
 3. The Cheese Shop
- 1

These are the contents stored in `COMP9001.txt` at the end of the game.

```
Start game
Start shop
Bought 1 cheddar
Bought 2 cheddar
Bought 3 cheddar
Bought 4 cheddar
Bought 0 cheddar
End shop
Start hunt
Caught a Brown mouse!
Caught a Brown mouse!
Caught a Brown mouse!
Caught a Brown mouse!
Nothing happens.
Caught a Brown mouse!
Nothing happens.
Caught a Brown mouse!
Caught a Brown mouse!
Nothing happens.
Nothing happens. You are out of cheese!
Nothing happens. You are out of cheese!
End hunt
Start shop
Bought 9 cheddar
Bought 10 cheddar
Bought 11 cheddar
Bought 12 cheddar
Bought 13 cheddar
End shop
Start hunt
Caught a Brown mouse!
Nothing happens.
Nothing happens.
Caught a Brown mouse!
```

```
Caught a Brown mouse!
Nothing happens.
Caught a Brown mouse!
Caught a Brown mouse!
Nothing happens.
Nothing happens.
Nothing happens.
Caught a Brown mouse!
Nothing happens.
Nothing happens.
Nothing happens.
Nothing happens.
Caught a Brown mouse!
Nothing happens.
Nothing happens.
Do nothing
Nothing happens.
Nothing happens.
Caught a Brown mouse!
Caught a Brown mouse!
Caught a Brown mouse!
Caught a Brown mouse!
Caught a Brown mouse!
Nothing happens.
Nothing happens.
Caught a Brown mouse!
Caught a Brown mouse!
End hunt
Start shop
End shop
End game
```

Question 2 - Game Analysis (35 Marks)



Write your answer for this question into fe.py.

Players can check if their account exists by passing the chosen in-game name as command line arguments. If insufficient arguments are received, the game displays the format of the program execution: `Format: python3 fe.py <name>` to inform players of the expected way to run their program. If the argument passed into the function is not a file object, the function does nothing and returns an empty string. If many command line arguments are received, only the first argument is used for checking. If a file with the in-game name is found, the program lists an analysis of the events that happened the last time the user played the game. Else, the program does nothing again and returns an empty string.

Valid game events in the file must be enclosed within the lines `Start game` and `End game`. Events that occur outside these lines are invalid events.

```
<Invalid event>
Start game
<Valid event 1>
<Valid event 2>
.
.
End game
<Invalid event>
```

The events that occur during the Hunt and within the shop must be within the lines `Start <place>` and `End <place>` where place must be `hunt` or `shop` respectively for it to be included in the count during the analysis. Events that occurs outside these fences are ignored. In the given example below, the line `Bought 10 cheddar` is the only valid event that occurs within the file.

```
Bought 5 cheddar
Start game
Start shop
Bought 10 cheddar
End shop
Bought 100 cheddar
End game
```

In the sample file `missing_end.txt`, the program should not find any valid events because the `End game` fence is missing. Hence, there are no valid events found in the file.

```
Start shop
End shop
Start game
Start shop
Bought 2 cheddar
End shop
Start hunt
Caught a Brown mouse!
Nothing happens.
Nothing happens. You are out of cheese!
Nothing happens. You are out of cheese!
End hunt
```

In the sample file `outside.txt`, the events on Line 5, 22 and 24 are invalid events because they occur outside the `game` and respective place fences.

During the analysis, the program must extract the following information from the valid events found in the text file and save them into the given variable names:

- Total cheese bought in The Cheese Shop stored in `total_cheddar`
- Total gold spent in the The Cheese Shop stored in `total_spent`
- Total times the hunter have sounded the horn stored in `total_sound`
- Total Brown mouse caught stored in `total_brown`
- Total number of times the trap is empty during a hunt stored in `total_misses`

- Total number of times they hunted without cheese stored in `total_forgot`
- Total revenue the hunter gained from The Hunt stored in `total_revenue`

A formatted string summarizing the player's events during the last game play is then displayed to terminal *exactly* as follows:

```
You have spent {total_spent} gold in The Cheese Shop.
Total cheddar bought: {total_cheddar}
You have sounded the horn {total_sound} times during The Hunt.
Total Brown mouse caught: {total_brown}
Total empty catches: {total_misses}
Total hunt without cheese: {total_forgot}
You have earned {total_revenue} gold from The Hunt.
```

Note: The first line is `You have spent {total_spent} gold in The Cheese Shop.` The last line is `You have earned {total_revenue} gold from The Hunt.` The statements that starts with `Total` have a single whitespace at the start of the sentence.

Write function `analyze_game` which has a single parameter representing the file object in `read` mode and returns a formatted string of the result of analysis. If an invalid argument is received by the function, the function returns an empty string. If a valid argument is received, the function analyzes the file object to extract the valid events that happens and produces a summary of the events. The result of analysis must be stored in the appropriate variable names as given above. The function should not display anything to terminal during the analysis. The function assumes the following while doing its analysis:

- A cheddar costs `10` gold.
- A Brown mouse will always drop `125` gold when caught.
- `total_misses` is the summation of the total times that the valid events of `Nothing happens.` occur.
- `total_forgot` is the summation of the total times that the valid events of `Nothing happens. You are out of cheese!` occur.
- `total_sound` is a summation of `total_brown`, `total_misses` and `total_forgot`.

This is example of the output produced when the file object for `outside.txt` is passed into the function:

```
You have spent 100 gold in The Cheese Shop.
Total cheddar bought: 10
You have sounded the horn 12 times during The Hunt.
Total Brown mouse caught: 2
Total empty catches: 8
Total hunt without cheese: 2
You have earned 250 gold from The Hunt.
```

Write function `main` which accepts a list of argument values received from the command line, `args`, and returns a `str` object representing the result of game analysis. The function assumes that the

data for past saved games will always be stored as a text file (`txt` extension) in the directory `/home/saved/` and labelled as the player's name. The function will attempt to read the file from this directory. If this read is unsuccessful, the function does nothing and returns an empty string. If it is successful, it then calls function `analyze_game` to analyze the contents of the file, displays the results of analysis and returns the result of analysis. If insufficient arguments are received by the program, the program displays the following message: `Format: python3 fe.py <name>` to the terminal. The main function should execute automatically when the script `fe.py` is run.

Program Execution

The auto-marker will only start the program `fe.py`. These are examples of the output produced for different command line arguments:

- Past game analysis for `Tom`

```
$ python3 fe.py Tom
You have spent 50 gold in The Cheese Shop.
Total cheddar bought: 5
You have sounded the horn 8 times during The Hunt.
Total Brown mouse caught: 3
Total empty catches: 2
Total hunt without cheese: 3
You have earned 375 gold from The Hunt.
```

- Past game analysis for `COMP9001`

```
$ python3 fe.py COMP9001
You have spent 650 gold in The Cheese Shop.
Total cheddar bought: 65
You have sounded the horn 42 times during The Hunt.
Total Brown mouse caught: 21
Total empty catches: 19
Total hunt without cheese: 2
You have earned 2625 gold from The Hunt.
```

- Invalid cases where `bilby` and `tom` does not exist

```
$ python3 fe.py bilby
$ python3 fe.py tom and jerry
```

- No command line arguments is given

```
$ python3 fe.py
Format: python3 fe.py <name>
```

- Many command line arguments, similar to checking for `Tom`.

```
$ python3 fe.py Tom and Jerry
You have spent 50 gold in The Cheese Shop.
```

Total cheddar bought: 5
You have sounded the horn 8 times during The Hunt.
Total Brown mouse caught: 3
Total empty catches: 2
Total hunt without cheese: 3
You have earned 375 gold from The Hunt.