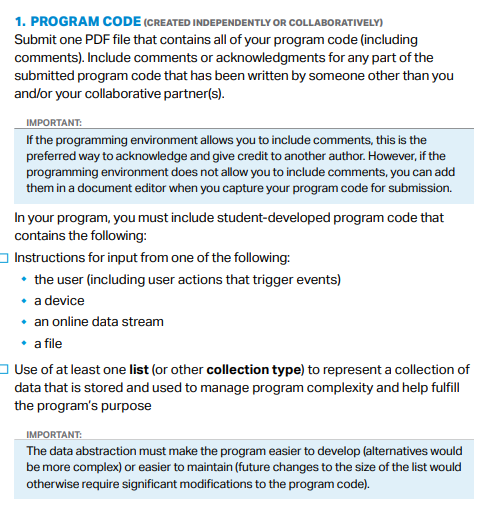
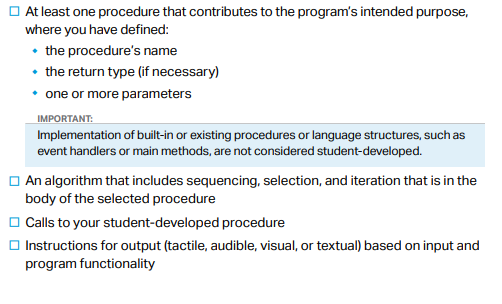
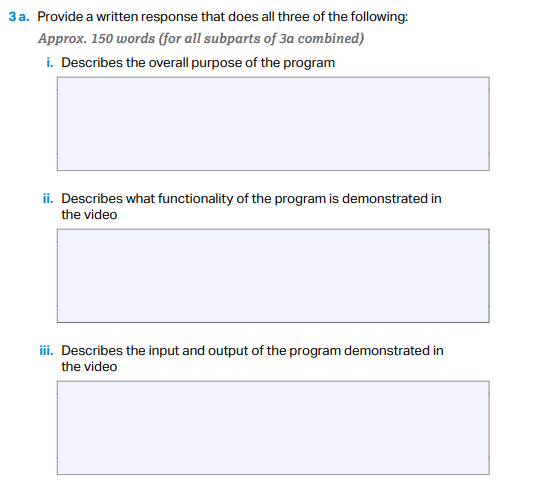
End of Semester Project:

In either SNAP or Python write a program and submit the following:



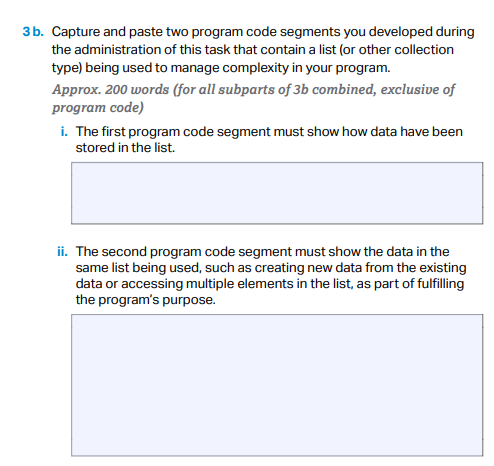


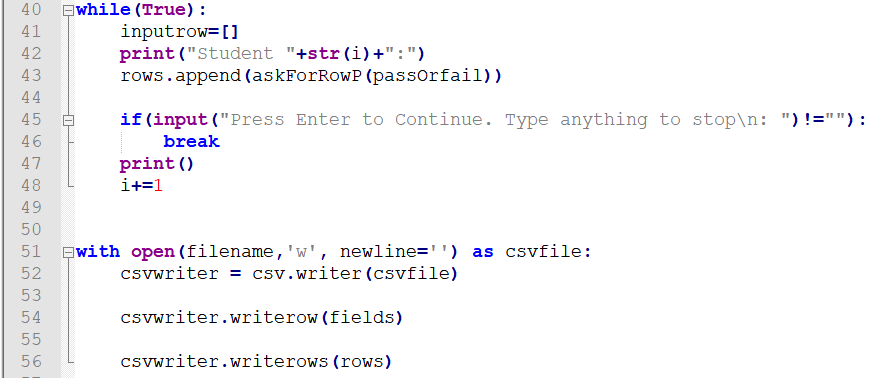


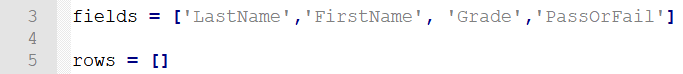
The functionality of the program in the video is to convert user inputs into a csv file. In the video, the program will ask the user for the student name, grade, and then if applicable it will determine whether if the student passed it or not. And then a csv file will appear in the same location as the grogram in the folder after the program ended.

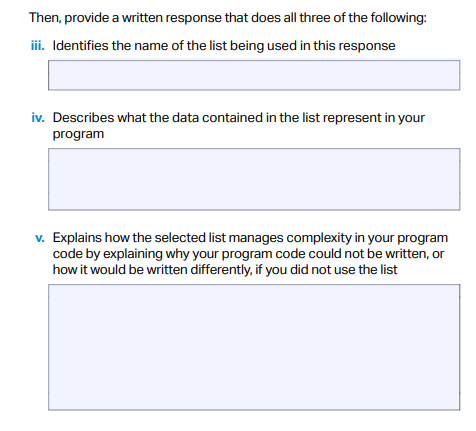
The input from the user will be the passing grade for the class, student last and first name, student’s numerical grade, and the indicator of whether the program should end or stop. The output will be a csv file that created to save the input data values.

The overall purpose of the program is to make a csv (comma separated values) for grades after receiving inputs from the user. This program is intended to save time (not sure if it really does it tho, 😊) and make grade inputting more enjoyable.





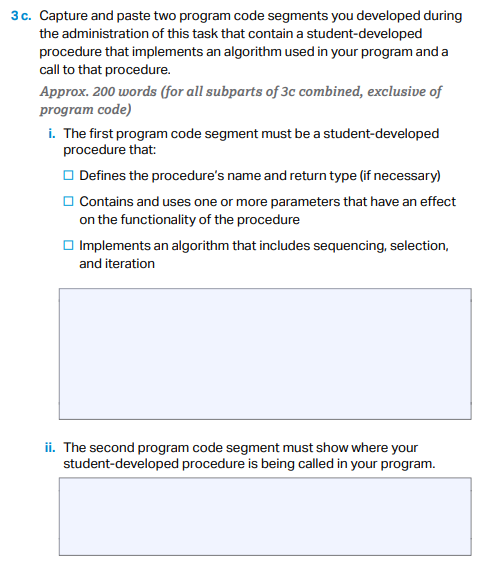
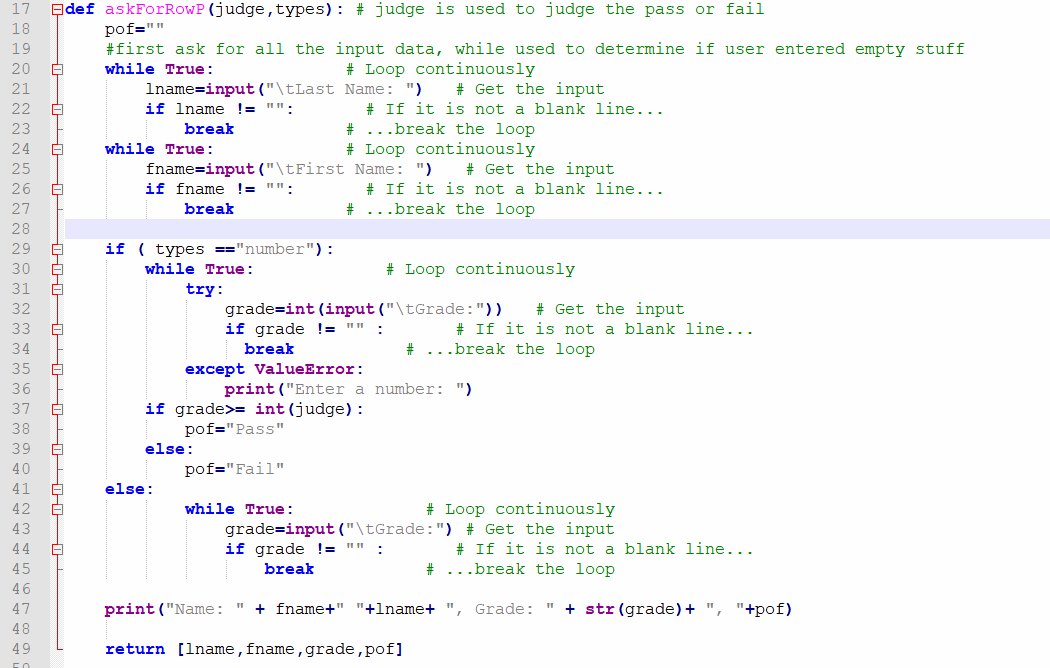


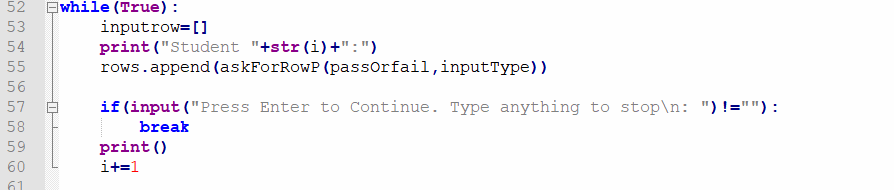


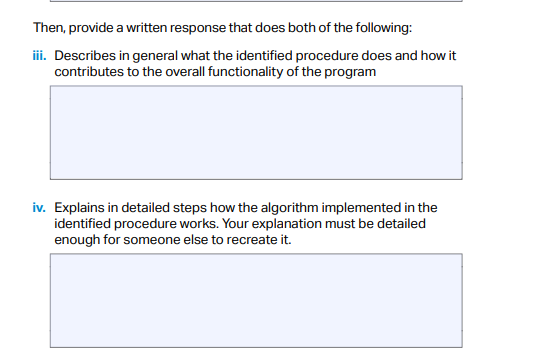
This list manages complexity in my program because I don’t need to create individual list for each of the students. Creating individual list for each of the students will be really challenging especially because the program don’t have a finite number of students; I can’t create an exact number of list that I will use to store the data later, therefore it will be challenging since that requires much more knowledge in computer to know how to initializing different variables names and use them conveniently in a loop.

It is initially an empty list it will later be use to store lists(users input). Each element(or sub-list) inside rows will represent one student in the class who has a grade.

The list name is *rows*.







1. As the function was called upon, user need to pass two variables to the program. One is the judge(the passing line) and the other one is the types(tell if it is numerical value).

2. I first created a variable *pof* and assigned it as an empty string for later usage in determining pass or fail status of the student.

2. I create 2 **while loops** to gather input and I will check if the input has something. I don’t accept empty input.

The while loops for *Last name* and *First Name* will continue running **if** the user’s input is empty, in other words equals to nothing(=””)—or, in other words, the loop will **break** if the input is not empty.

3. For grade, first I will use a **if statement** to determine if I should have the step to compare the **grade** with the value stored in *judge*.

If I know the inputType is “number”: I will ask user for numerical inputs. So that when I need to **cast** the string input to a integer, it won’t give back errors. SO I decide to use **try except** to make sure the program will send out a error message to tell user “to reenter the number:” and let the while continue loop until there is **no ValueError** and the input is not empty. And the program will do an **if statement** to check if the grade is a passing grade(>= to parameter value *judge*), and change the value of *pof* variable accordingly.

Else I will just do the same as I did in asking first name and last name. Just ask about it and make sure it is not empty.

4. Before I return the output of the function, I print the entered data values to led the user know what he/she had inputted.

5. Return all the values in a list ([] in square brackets) the order like [LastName, FirstName, Grade, PassOrFail].

askForRowP() is a function that is used to ask user for three different inputs: last name, first name, and grade. It will then return an array that contains all three inputs and the *PassOrFail* status in consecutive order. It helps the program to gather information for individual student, and return it as a list.

Provide a Written response that does all here of the following.

i. Describe two calls to the procedure identified in written response 3c. Each call must pass a different argument(s) that causes a different segment of code in the algorithm to execute.

First call:



Assume that user’s input correspond to passOrfail will be 65 and the inputType is specified to be “number”.

*n*

Second call



Assume that user’s input correspond to passOrfail will be 65 and the inputType is specified to something **not** “number”.

ii. Describe what conditions(s) is being tested by each call to the procedure.

Condition(s) tested by the first call:

Since the program now know that the input for grades will be “number”, it will make sure that the user inputted numerical numbers and the input is not empty. It will also make a judgement on whether the student passed or failed the test. And the value for *pof* will be either “pass” or “fail".

Conditions(s) tested by the 2nd call:

Since the program now know that the input for grades will not be “number”, it won’t make sure user did input numerical numbers and it will not decide whether the student passed or failed the test, but indeed the value for *pof* will remain empty.

iii. Identifies the result of each call.

Result of first call:

The returned list will have have a value either is “pass” or “fail” as the fourth element(index = 3) in the list.

Ex: if passOrFail=65: [“li”,”nan”,”65”, “pass’’]

Result of Second call:

The returned list will have have a value of “”(an empty string) as the fourth element(index = 3) in the list.

Ex: if passOrFail=65: [“li”,”nan”,”65”, “ ’’]

Ex2 : if passOrFail= B : [ “li”,”nan”,”A’’, “ ’’]