**IF100 – Fall 2021-2022**

**Take-Home Exam 3   
Due December 10th, 2021, Friday, 23:59 (Sharp Deadline)**

**Introduction**

The aim of this take-home exam is to practice on loops (for/while statements). The use of loops is due to the nature of the problem; that is, you cannot finish this take-home exam without using loop statements.

**Description**

Assume that you are a software engineer who tries to lose weight. You went to the nutritionist and asked him/her for help. Luckily, (s)he scheduled the amount of calories you can take per day by eating only the foods in a given list. Since you are a software engineer, you wanted to write a program that will keep you on track. By using this program, you want to choose a meal that will not exceed the daily calorie threshold. If the chosen meal exceeds the threshold, then the program will suggest another meal that is in the same group with the chosen meal that does not exceed the threshold (if there are any).

In this take-home exam, you will implement a Python program that will get four (4) inputs from the user:

* The first input to your program is the food list. This list contains the names of the foods (meal), the groups that foods belong to and the amounts of calories per portion size. Each element of this list is separated by a comma character (,), and each information for a particular element is separated by a colon character (:).
* The second input to your program is the food(s) you ate so far during the day. This input will contain only the name(s) of the food(s), where each meal is separated by a comma character (,) if there are multiple of them.
* The third input to your program is the amount of calories you can take during the day.
* The fourth input to your program is the food you want to eat next.

Your program needs to print an appropriate output according to some conditions, which are listed below.

* If (any of) the food(s) entered in the second input (the things you ate so far) is not in the list entered as the first input (the food list), then your program should print out an appropriate message. That is, you need to print different error messages for each food that does not exist in the database.
* If the food you want to eat (fourth input) is not in the list entered as the first input, then your program should print out an appropriate message.
* If the food you want to eat (fourth input) results in exceeding the threshold (designated by third input) once you eat it, *and* there is no other food in the same group (specified by the first input) that will not cause a calorie amount which is greater than the threshold after it is eaten, then your program should print out an appropriate message.
* If the food you want to eat (fourth input) results in exceeding the threshold (designated by third input) once you eat it, *and* there is another food in the same group (specified by the first input) that will not cause a calorie amount which is greater than the threshold after it is eaten, then your program should print out an appropriate message.
* If the total calories you take after you eat the food that you want (fourth input) does not exceed the threshold (designated by third input), then your program should print out an appropriate message.

You can find the details about the inputs and outputs in the following section.

**Input, Process and Output**

The inputs of the program and their order are explained below. It is extremely important to follow this order with the same format since we automatically process your programs. Also, prompts of the input statements to be used have to be exactly the same as the prompts of the "Sample Runs". ***Thus, your work will be graded as 0 unless the order is entirely correct***.

Your program will have multiple inputs and the specifications for these inputs are explained below.

* *Food List*: First comes the name of the food (meal), and then comes the group that the food belongs to, and at last, there is the amount of calorie per portion size in the following format:

***food#1:group#1:calories#1,food#2:group#2:calories#2,...,food#N:group#N:calories#N***

You may assume that the database will be given correctly and there will be no colon ("**:**") and comma ("**,**") characters used in neither food nor group names.

* + You may assume that there will be only one colon (":") between each information and only one comma (",") between the elements of the list.
  + There will not be any colon, or comma characters in the beginning or in the end of the input.
  + You don't need to perform any input checks for this input.
* *Food Eaten So Far*: This input will include the foods you ate so far, in a comma separated way, and it will be in the following format:

***food#1,food#2,...,food#N***

* + You may assume that the user will not enter spaces after each comma.
  + The user may also enter only one food. In such a case, the comma character will not be used.
  + There will not be any comma (,) characters in the beginning or in the end of the input.
  + You may assume that there will be only one comma (,) between each food.
* *Calorie Amount*: This input provides information on the amount of calories you can take during the day.
  + You may assume that the user will enter a positive real value.
* *What to Eat Next*: This input provides information on the food you want to eat next.

After you get all of the inputs, your program needs to perform a validity check to make sure that the entered foods (within the second input in a **case-insensitive** way) exist in the food list. If not, then, your program should output;

**"***food* **is not in the dataset."**

This validity check must be done for each food available in the second input, ***individually***.

The third and fourth inputs are the total calories you can take in a day and the food you want to eat next, respectively. If the second input is valid totally, then you will do another validity check. If the willing food does not exist in the food list (which is case insensitive), then your program should output;

**"***willing\_food* **does not exist in the dataset."**

If the willing food exists in the food list, but the total calories you take after eating this food becomes greater than or equal to the total calories you can take in a day, then your program will look to the database (food list) to check if there is a food that will not lead you to have greater than or equal to the total calories you can take in a day after it is eaten, *as of the same group* with the willing food. If there is, then your program should output;

**"You cannot eat** *willing\_food***. However, we are suggesting you to eat** *suggested\_food***."**

Keep in mind that the suggested food will be chosen as the **first food** that stands in the food list as well as in the same group and will not cause exceeding the limit with the willing food.

If there is no such food, then your program should output;

**"There is no food you can eat from** *willing\_foods\_group* **group."**

If the chosen food does not exceed the total calories you can take in a day after it is eaten, then your program should output;

**"You can eat** *willing\_food* **that you will get** *calories\_you\_took* **calories."**

If you find any validity mistake in any input, then you should not make a validity check of the remaining inputs (you should still get them from the user in either case, but will not check their validity). While checking the validity of the second input, you need to find all erroneous cases for each food given in this input, and you should not make a validity check in the rest of the inputs again. Additionally, in all possible cases, the food names and groups should be printed in the original way (in the way that they inputted). Please see the "Sample Runs" section for some examples.

**Sample Runs**

Below, we provide some sample runs of the program that you will develop. The *italic* and **bold** phrases are inputs taken from the user. You have to display the required information in the same order and with the same words and characters as below. **Sample Run 1**

Please enter the dataset you have: Jam:**Fats & Sugars:38cals,Bagel:BREADS & CEREALS:140cals,Beef (roast):Meats & Fish:300cals,Liver pate:Meats & Fish:150cals,Corn snack:Fats & Sugars:125cals,Jaffa cake:BREADS & CEREALS:48cals,Syrup:Fats & Sugars:15cals,Cockles:Meats & Fish:50cals,Mars bar:Fats & Sugars:240cals,Chapatis:BREADS & CEREALS:250cals,Ham:Meats & Fish:6cals,Crackerbread:BREADS & CEREALS:17cals,Rice (egg-fried):BREADS & CEREALS:500cals,Fish cake:Meats & Fish:90cals,Salmon fresh:Meats & Fish:220cals,Toffee:Fats & Sugars:100cals,Steak & kidney pie:Meats & Fish:400cals**

PLease enter the food(s) you ate during the day: **Jam,Liver,Corn snack**

Please enter the amount of calories you can take in a day: **1200**

Please enter the food you want to eat: **Rice (egg-fried)**

Liver is not in the dataset.

**Sample Run 2**

Please enter the dataset you have: **Jam:Fats & Sugars:38cals,Bagel:BREADS & CEREALS:140cals,Beef (roast):Meats & Fish:300cals,Liver pate:Meats & Fish:150cals,Corn snack:Fats & Sugars:125cals,Jaffa cake:BREADS & CEREALS:48cals,Syrup:Fats & Sugars:15cals,Cockles:Meats & Fish:50cals,Mars bar:Fats & Sugars:240cals,Chapatis:BREADS & CEREALS:250cals,Ham:Meats & Fish:6cals,Crackerbread:BREADS & CEREALS:17cals,Rice (egg-fried):BREADS & CEREALS:500cals,Fish cake:Meats & Fish:90cals,Salmon fresh:Meats & Fish:220cals,Toffee:Fats & Sugars:100cals,Steak & kidney pie:Meats & Fish:400cals**

PLease enter the food(s) you ate during the day: **Jam,Liver,Cornn snack**

Please enter the amount of calories you can take in a day: **1200**

Please enter the food you want to eat: **Rice (egg-fried)**

Liver is not in the dataset.

Cornn snack is not in the dataset.

**Sample Run 3**

Please enter the dataset you have: **Jam:Fats & Sugars:38cals,Bagel:BREADS & CEREALS:140cals,Beef (roast):Meats & Fish:300cals,Liver pate:Meats & Fish:150cals,Corn snack:Fats & Sugars:125cals,Jaffa cake:BREADS & CEREALS:48cals,Syrup:Fats & Sugars:15cals,Cockles:Meats & Fish:50cals,Mars bar:Fats & Sugars:240cals,Chapatis:BREADS & CEREALS:250cals,Ham:Meats & Fish:6cals,Crackerbread:BREADS & CEREALS:17cals,Rice (egg-fried):BREADS & CEREALS:500cals,Fish cake:Meats & Fish:90cals,Salmon fresh:Meats & Fish:220cals,Toffee:Fats & Sugars:100cals,Steak & kidney pie:Meats & Fish:400cals**

PLease enter the food(s) you ate during the day: **Crackerbread,Cockles,Toffee,Salmon fresh**

Please enter the amount of calories you can take in a day: **1500**

Please enter the food you want to eat: **Cornn snack**

Cornn snack does not exist in the dataset.

**Sample Run 4**

Please enter the dataset you have: **Jam:Fats & Sugars:38cals,Bagel:BREADS & CEREALS:140cals,Beef (roast):Meats & Fish:300cals,Liver pate:Meats & Fish:150cals,Corn snack:Fats & Sugars:125cals,Jaffa cake:BREADS & CEREALS:48cals,Syrup:Fats & Sugars:15cals,Cockles:Meats & Fish:50cals,Mars bar:Fats & Sugars:240cals,Chapatis:BREADS & CEREALS:250cals,Ham:Meats & Fish:6cals,Crackerbread:BREADS & CEREALS:17cals,Rice (egg-fried):BREADS & CEREALS:500cals,Fish cake:Meats & Fish:90cals,Salmon fresh:Meats & Fish:220cals,Toffee:Fats & Sugars:100cals,Steak & kidney pie:Meats & Fish:400cals**

PLease enter the food(s) you ate during the day: **Steak & kidney pie,Chapatis,Corn snack**

Please enter the amount of calories you can take in a day: **780**

Please enter the food you want to eat: **Beef (roast)**

There is no food you can eat from Meats & Fish group.

**Sample Run 5**

Please enter the dataset you have: **Jam:Fats & Sugars:38cals,Bagel:BREADS & CEREALS:140cals,Beef (roast):Meats & Fish:300cals,Liver pate:Meats & Fish:150cals,Corn snack:Fats & Sugars:125cals,Jaffa cake:BREADS & CEREALS:48cals,Syrup:Fats & Sugars:15cals,Cockles:Meats & Fish:50cals,Mars bar:Fats & Sugars:240cals,Chapatis:BREADS & CEREALS:250cals,Ham:Meats & Fish:6cals,Crackerbread:BREADS & CEREALS:17cals,Rice (egg-fried):BREADS & CEREALS:500cals,Fish cake:Meats & Fish:90cals,Salmon fresh:Meats & Fish:220cals,Toffee:Fats & Sugars:100cals,Steak & kidney pie:Meats & Fish:400cals**

PLease enter the food(s) you ate during the day: **Mars bar,Bagel,Fish cake**

Please enter the amount of calories you can take in a day: **500**

Please enter the food you want to eat: **Toffee**

You cannot eat Toffee. However, we are suggesting you to eat Syrup.

**Sample Run 6**

Please enter the dataset you have: **Jam:Fats & Sugars:38cals,Bagel:BREADS & CEREALS:140cals,Beef (roast):Meats & Fish:300cals,Liver pate:Meats & Fish:150cals,Corn snack:Fats & Sugars:125cals,Jaffa cake:BREADS & CEREALS:48cals,Syrup:Fats & Sugars:15cals,Cockles:Meats & Fish:50cals,Mars bar:Fats & Sugars:240cals,Chapatis:BREADS & CEREALS:250cals,Ham:Meats & Fish:6cals,Crackerbread:BREADS & CEREALS:17cals,Rice (egg-fried):BREADS & CEREALS:500cals,Fish cake:Meats & Fish:90cals,Salmon fresh:Meats & Fish:220cals,Toffee:Fats & Sugars:100cals,Steak & kidney pie:Meats & Fish:400cals**

PLease enter the food(s) you ate during the day: **Steak & kidney pie,Liver pate,Corn snack**

Please enter the amount of calories you can take in a day: **1500**

Please enter the food you want to eat: **Rice (egg-fried)**

You can eat Rice (egg-fried) that you will get 1175 calories.

**What and where to submit?**

You should prepare (or at least test) your program using Python 3.x.x. We will use Python 3.x.x while testing your take-home exam. Let us repeat,

* You must use Google Colab to develop your code from scratch (from beginning till the end), and then submit it **through SUCourse+ only**! Once you are done with developing your code on Google Colab, then you will copy your code to the CodeRunner to see if your program can produce the correct outputs. At the end, you will submit your code through CodeRunner (and SUCourse+). You should keep your Google Colab file until the end of the semester, we might want to look at this. If you fail to provide this Google Colab file anytime in the semester, you may not earn any credits from this Take Home Exam.
* In the CodeRunner, there are some visible and invisible (hidden) test cases. You will see your final grade (including hidden test cases) before submitting your code. Thus, it will be possible to know your THE grade before submitting your solution.
* **There is no re-submission**. You don't have to complete your task in one time, you can continue from where you left last time but you should not press submit before finalizing it. Therefore, you should make sure that it’s your final solution version before you submit it.
* **General Take-Home Exam Rules**
* Successful submission is one of the requirements of the take-home exam. If, for some reason, you cannot successfully submit your take-home exam and we cannot grade it, your grade will be 0.
* There is NO late submission. You need to submit your take-home exam before the deadline. Please be careful that SUCourse+ time and your computer time may have 1-2 minutes differences. You need to take this time difference into consideration.
* Do NOT submit your take-home exam via email or in hardcopy! SUCourse+ is the only way that you can submit your take-home exam.
* If your code does not work because of a syntax error, then we cannot grade it; and thus, your grade will be 0.
* Please submit your **own** work only. It is really easy to find "similar" programs!
* Plagiarism will not be tolerated. Please check our plagiarism policy given in the syllabus of the course.

Good luck!

Özgün Yargı

& IF100 Instructors