PROGRAMMING ASSIGNMENT 3

TAs : Nebi YILMAZ
Due Date : 08.12.2022 (23:00)
Click here to accept your Programming Assignment 3.

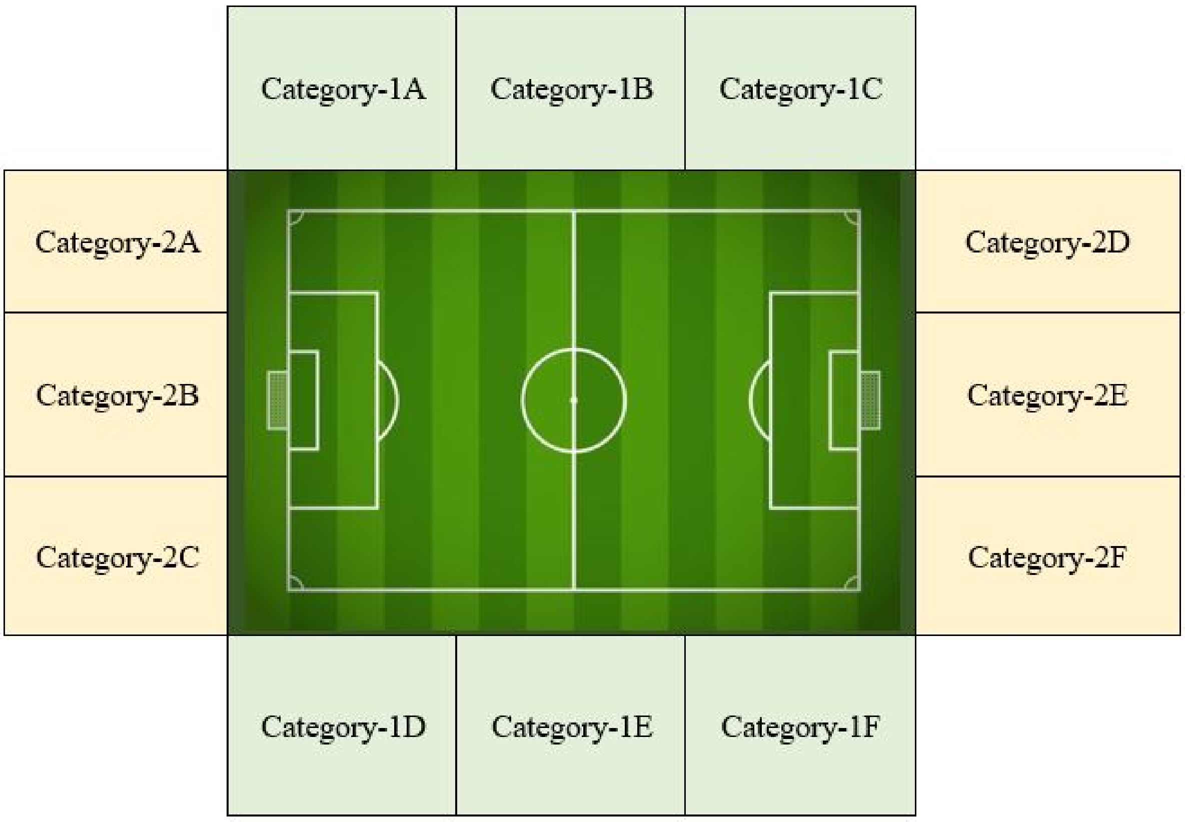

Figure 1: Overview of the Şükrü Saraçoğlu Stadium

1 Assignment

In this assignment, you will get familiar with file operations, lists, and dictionaries while you implement a real-world Football ticketing system. Furthermore, you will have hands-on experience designing required data structures to store related data for categories and seats in the Şükrü Saraçoğlu Stadium while you do some operations such as selling or canceling tickets. As can be seen in Figure 1, seats for football fans consist of categories and ticket transactions are made through these categories. There are two type of category as category-1 (i.e., Marathon tribune) and category-2 (i.e., back-goal tribune). Each category (e.g., category-1) can be divided into more than one category (e.g., category-1A, category-1B, etc.) in itself.

You will be supposed to design and implement a console-based basic Football ticketing system that reads an input file as an argument line by line and executes each line contains a command and its arguments. The name of the input file will be fixed as "**input.txt**" and it will be inside the folder where your python file is located. A list of available commands is given in Section 3. In our stadium, arbitrary number of categories can be created with a rectangular layout having the columns named with a sequence of 0, 1, 2, 3, ... while its rows are named with the characters of the English alphabet (A, B, C, ..., Z). In Figure 2, this scheme has been illustrated. **As shown the figure, the side of each category facing the football ground will be the columns and the other side will be the rows**. As a result, the number of rows of any category is limited to 26. In our stadium, you should apply three types of fares for each seat (i.e. student, full pay or season ticket). According to these tariffs, the price of student and full fare tickets will be 10 and 20 dollars, respectively. The price of a season ticket will be 250 dollars. In the following sections, you will find the descriptions of the commands and their arguments. Before delving into details please note that your program must output the results of the operations to both a console and a text file. The name of the output file will be fixed as "**output.txt**" and it must be created in the same folder where your python file exists.

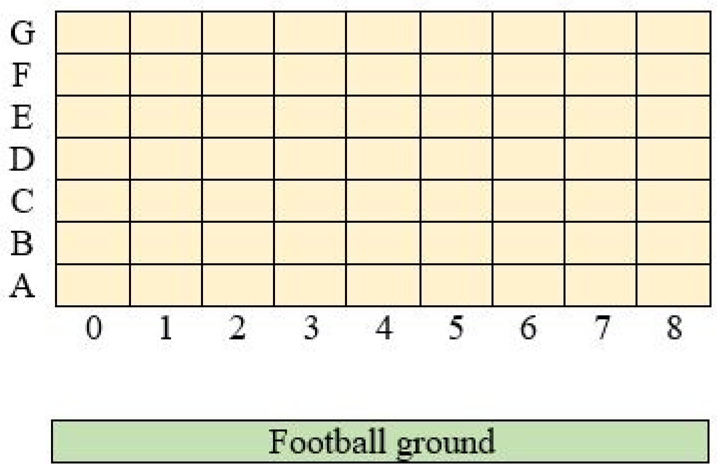

Figure 2: A sample category with its rows and columns

2 Commands

You are supposed to write a Python program to read an input file and execute commands inside of it. The name of the input file must be accepted as an argument by your program. The name of your program must be "assignment3.py". The input files will reside in the directory your python file exists. A list of the commands is given in the following table, and their details are given in the following sub-sections.

List of command	Explanation
CREATECATEGORY	This command is used to create a category consisting of rows and columns for football fans
SELLTICKET	This command is used to sell tickets to football fans from the category they specify.
CANCELTICKET	This command is used to cancel the tickets that football fans have purchased from the category they have determined.
BALANCE	This command shows the number of tickets sold by type of football fans in a certain category and the total revenue in that category
SHOWCATEGORY	This command is used to visualize the current layout of the specified category with their seats and their actual status.

2.1 CREATECATEGORY

This command expects two arguments to create a free category. Initially, all the seats must set to free. The syntax of the command is given below:

```
CREATECATEGORY categoryname number_of_rowsXnumber_of_columns
```

Example command:

```
CREATECATEGORY category-1D 20x15
```

To this end, when you read such a line you must create a category which will be called "category-1D" and has 20 rows and 15 columns resulting in 300 total seats. If the operation is successfully carried out, you must output a success prompt like below:

```
The category 'category-1D' having 300 seats has been created
```

As shown in Figure 1, the category names are unique. However, note that, once the category is defined it will be no more possible to redefine it. So, your program must output an error message if you come across another "CREATECATEGORY" command involving the same name you have already assigned. The example error message is presented below:

```
Warning:  Cannot create the category for the second time.  The stadium
has already category-1D.
```

2.2 SELLTICKET

The command "SELLTICKET" enables you to sell tickets. This command fundamentally takes four arguments. However, it may take an arbitrary number of arguments for multiple seats or seat ranges. The basic syntax is given below:

```
SELLTICKET customer_name full|student|season category_name seat*
```

As can be seen, the first argument indicates the customer name whereas the second one identifies the payment type. As told before, tickets are categorized into 3 groups which are either "seasons", "full" or "student". The third argument stands for the target category name in which the seat or seats will be purchased. The last argument (may have more than one) specifies the seat (e.g. B13), seats (e.g. B13 B14 A20 C2), or seat ranges (e.g. D2-15). Thus, you must deal with a variational number of command arguments and process each of them separately. Some command examples are given below:

```
SELLTICKET hagi full category-1D B12
SELLTICKET alex student category-1D B12 C3 B5 D4

SELLTICKET maria student category-1E C2-4 D7 A6

SELLTICKET nihat full category-1E C3-15

SELLTICKET neslihan student category-1D D7-32
```

Note that, your input file will involve lots of "SELLTICKET" commands each having a unique customer name. At this point, your algorithm must handle all command arguments separately and check whether the respective seat or seat ranges are available to sell. As a rule, any command argument indicating an individual seat (e.g. B12) should be checked for its availability whereas all the seats in a range must be completely checked before their purchase. In other words, you cannot sell a seat range even one of the seats in it is not available. As a result, the output of each "SELLTICKET" command involves one or more than one lines each indicating the result of each seat or seat range. Meanwhile, you must also check the category name and number of command arguments against any exceptional case. Moreover, seat ranges or individual seats may exceed the category sizes. For instance, if the seat label (e.g. C30) does not exist in the related target category, your program must create a warning for that operation. Some example output lines are given below with respect to the example commands above:

```
Success:  hagi has bought B12 at category-1D

Warning:  The seat B12 cannot be sold to alex since it was already sold!

Success:  alex has bought C3 at category-1D

Success:  alex has bought B5 at category-1D

Success:  alex has bought D4 at category-1D

Success:  maria has bought C2-4 at category-1E

Success:  maria has bought D7 at category-1E

Success:  maria has bought A6 at category-1E

Error:  The seats C3-15 cannot be sold to nihat due some of them have
already been sold!

Error:  The category 'category-1D' has less column than the specified
index D7-32!
```

2.3 CANCELTICKET

The command "CANCELTICKET" enables you to cancel some purchases. It has a varying number of arguments such as "SELLTICKET". The basic syntax of the command is given below:

```
CANCELTICKET category_name seat*
```

Similar to the "SELLTICKET" command, it takes an arbitrary number of command arguments and tries to cancel the operation and revert the status of each seat to which each argument points. Unlike "SELLTICKET", the last argument (may have more than one) specifies the seat (e.g. B13) or seats (e.g. B13 B14 A20 C2), but not specifies the seat ranges (e.g., D2-15). The same exceptional cases hold for ”SELLTICKET” also exist for ”CANCELTICKET”. Furthermore, if the command argument points to any seat which has never been sold, you must output a warning. Some example commands and their respective outputs are given in an ordered way below:

```
CANCELTICKET category-1D B12

CANCELTICKET category-1E C3 B5

CANCELTICKET category-1F E33

CANCELTICKET category-1D G45
```

Output:

```
Success:  The seat B12 at ’category-1D’ has been canceled and now ready
to sell again

Success:  The seats C3 at ’category-1E’ have been canceled and now ready
to sell again

Error:  The seat B5 at ’category-1E’ has already been free!  Nothing
to cancel

Error:  The category ’category-1F’ has less column than the specified
index D33!

Error:  The category ’category-1F’ has less column than the specified
index G45
```

2.4 BALANCE

The command ”BALANCE” is used to get a balance report of any category with its revenues. The command takes only the category names as arguments. The syntax of the command is given below:

```
BALANCE category_name
```

According to this definition some examples are listed below:

```
BALANCE category-1D
```

When your program executes this command it must first parse the command line and list the balance information involving student, full pay, season ticket and overall revenues. An example output is presented below:

```
Category report of ’category-1D’

--------------------------------

Sum of students = 5, Sum of full pay = 10, Sum of season ticket=2, and
Revenues = 750 Dollars
```

As you may observe, the line below the first line has the same string length as the header line of the command output.

2.5 SHOWCATEGORY

The "SHOWCATEGORY" command is used to visualize the current layout of the categories with their seats and their actual status. The command takes only one argument which is the category name. An instance of the command is given below:

```
SHOWCATEGORY category-1D
```

In this example output you see the category layout according to a category named "category-1D". Assume that this category has 10 rows and 10 columns. As can be seen, the rows are labeled with letters of the English alphabet, and they are ordered in descending fashion (i.e. O, N, M, L, ..., A). Likewise, the columns must be placed one after another having 2 white space character. The columns must be started from 0 to the count of the last column. In this visualization, the empty seats (X) are illustrated with the character of 'X' while the seats sold for students (S), full pay (F), and season tickets (T) can be shown with 'S', 'F', and 'T' respectively.

```
Printing category layout of category-1D

J X X S X X S X X X G
I X X G X X X G X X G
H X X X X X X S X X G
G X X X X X X G X X X
F X X X T X S S X X X
E X X X X X X G X X X
D X X X X X T G X X X
C X X X X G X X X X X
B X X X G X X X X X X
A X X X X X X X X X X
  0 1 2 3 4 5 6 7 8 9
```

You should keep in mind that your output must be formed and listed exactly the same as the example given above.

3 Grading Policy

Task	**Point**
Submit	1
Clean code	10
Coding standard	5
Output	84
Total	100

4 Important Notes

- Compile your code on `dev.cs.hacettepe.edu.tr` before submitting your work to make sure it compiles without any problems on our **DEV** server.

- The assignment must be original, individual work. Downloaded or modified source codes will be considered cheating. Also, the students who share their work will be punished in the same way.
- We will be using a software similarity program to identify the cases of possible plagiarism. Our department takes the act of plagiarism very seriously. Those caught plagiarizing (both originators and copiers) will be sanctioned.
- In the ”Coding style” of the grading policy section, it will be taken into account whether unnecessary code fragments are used whether the abstraction level is regular, and whether your code is understandable.
- In the ”Coding standard” of the grading policy section, it will be taken into account whether variable names or function names are used in the correct format.
- You can ask your questions through the course's piazza group and you are supposed to be aware of everything discussed in the piazza group. General discussion of the problem is allowed, but DO NOT SHARE answers, algorithms, source codes, and reports.
- Execute your code on the **DEV** server using the following command in terminal:

  python3 assignment3.py input.txt
- Don't forget to write comments on your codes when necessary.
- Save all work until the assignment is graded.
- Do not miss the deadline.
- You must submit your work with the file hierarchy as stated below:

  assignment3.py