CSE 231 **Spring 2023**

Computer Project #8

This assignment focuses on the design, implementation and testing of a Python program that uses Dictionaries and Sets.

It is worth 60 points (6% of course grade) and must be completed no later than **11:59 PM on Monday, April 10, 2023).** After the due date, 1 pt will be deducted for every 1 hr late. Note that you need to click on the submit button to be considered as submitted.

Assignment Overview

In this assignment, you will practice creating a dictionary and Set from a CSV file.

Assignment Background
Steam is a video game digital distribution service and storefront by Valve. It was launched as a software client in September 2003 as a way for Valve to provide automatic updates for their games and expanded to distributing and offering third-party game publishers' titles in late 2005.

Assignment Specifications

You will develop a Python program that makes use of steam data and answer user search queries by finding appropriate games from the database.

`Open_file (s) → file pointer:`

a. This function prompts the user to input a csv file name to open and keeps prompting until a correct name is entered. The parameter `s` is a string to incorporate into your prompt so you are prompting the user for a particular type of file (`"games"` , `"discount"`). Return the file pointer. Use try-except to determine if the file can be opened. Check the `strings.txt` file for the appropriate strings to use in the prompt and error message to match the tests. Use the `'UTF-8'` encoding when opening the file. For example:

```
fp = open("discount_small1.csv", encoding='UTF-8')
```

b. Parameters: `s (str)`
c. Returns: a file pointer
d. Display: prompts and error messages

`read_file (fp_games) → dictionary:`

a. This function uses the provided file pointer and reads the games data file. You need to use the csv module. The file has a header line that you need to skip. The file has the following columns: `name`, `release_date`, `developer`, `genres`, `player modes`, `price`, `overall_reviews`, `reviews`, `percent_positive`, `win_support`, `mac_support`, `lin_support`.
b. This function returns a dictionary with `name` of the game as the `key` and a list of the remaining data as the `value`. The dictionary will have the following structure:

```
{name: [date, [developer], [genre], mode, price, overall_reviews,
reviews, percent_positive, [support]]}
```

With the following format:

```
{str: [ str,list,list,int,float,str,int,int,list ]}
```

You will need to do the following conversions before adding the remaining data to the dictionary:

1. Convert `genres`, and `developer` to lists since there could be several of both which are separated by a semicolon (`";"`).
2. `player modes` column is a series of modes separated by a semicolon (`";"`). We are only interested in the first mode. If the first mode is `multi-player` (or any combination of cases, i.e., `Multi-player` or `Multi-Player, etc`…) then `mode` is `0`, otherwise it is `1`.
3. Convert `price` to a float. If the `price` is not a valid `float` then `price` will be set to 0.0. Remember to remove the comma from price before converting to a float. Also, the price of the game is in Indian rupees and needs to be converted to dollars using the rate:
   `1 rupee = 0.012 dollars.`
4. Convert `reviews` and `percent_positive` to integers. Remember that you need to remove `%` before converting `percent_positive` to integer.
5. Convert the last three columns to a list stating all the available supports. Support is available if the value is 1 for that support column. We have 3 supports possible: `win_support`, `mac_support`, and `lin_support`.

For example, if you use `games_clean_small.csv` file, then the dictionary will be:

```
{
'Grand Theft Auto V': ['13/04/2015', ['Rockstar North'], ['Action', 'Adventure'],
       1, 0.0, 'Positive', 349924, 78, ['win_support']],
'DayZ': ['13/12/2018', ['Bohemia Interactive'], ['Action', 'Adventure', 'Massively
       Multiplayer'], 0, 33.588, 'Positive', 55690, 87, ['win_support']],
'Bloons TD 6': ['17/12/2018', ['Ninja Kiwi'], ['Strategy'], 1, 5.268, 'Positive',
      442, 90, ['win_support', 'mac_support']],
'Counter-Strike: Global Offensive': ['21/08/2012', ['Valve', 'Hidden Path
      Entertainment'], ['Action', 'Free to Play'], 0, 0.0, 'Positive', 6774812, 88,
      ['win_support', 'mac_support', 'lin_support']],
'Dota 2': ['9/7/2013', ['Valve'], ['Action', 'Free to Play', 'Strategy'], 0, 0.0,
      'Positive', 1885261, 82, ['win_support', 'mac_support', 'lin_support']],
'Artifact': ['28/11/2018', ['Valve'], ['Strategy'], 1, 3.588, 'Positive', 309, 71,
      ['win_support']],
'SkiFy': ['24/01/2018', ['Blup Games'], ['Casual', 'Indie', 'Simulation'], 1,
      0.504, 'Mixed', 18, 55, ['win_support']]
}
```

c. Parameter : `a file pointer`
d. Returns : a `dictionary`
e. The function displays nothing.

`read_discount (fp_discount)` → `dictionary`:

a. The function would read the discount file and create a `dictionary` with <u>key</u> as the `name` of the game and <u>value</u> as the `discount` as a `float` rounded to 2 decimals. Remember to skip the header line.
b. Parameters: a `file pointer`
c. Returns: a `dictionary`
d. This function displays nothing.

For example, if you use `discount_small.csv` file, then the dictionary will be:

```
{"Tom Clancy's Rainbow Six Siege": 60.06, "No Man's Sky": 50.03, 'SCUM':
50.07, 'X4: Foundations': 60.03, 'SkiFy': 50.0, 'X3: Reunion': 53.0}
```

In_year (master_D,year) → list:

a. This function filters out games that were released in a specific `year` from the main dictionary you have created in the `read_file` function (`master_D`). Remember that `date` in `master_D` is of the form `DD/MM/YYYY`. The function returns a list of game names sorted alphabetically. For example:

```
master_D = {
'Grand Theft Auto V': ['13/04/2015', ['Rockstar North'], ['Action', 'Adventure'],
       1, 0.0, 'Positive', 349924, 78, ['win_support']],
'DayZ': ['13/12/2018', ['Bohemia Interactive'], ['Action', 'Adventure', 'Massively
       Multiplayer'], 0, 33.588, 'Positive', 55690, 87, ['win_support']],
'Bloons TD 6': ['17/12/2018', ['Ninja Kiwi'], ['Strategy'], 1, 5.268, 'Positive',
      442, 90, ['win_support', 'mac_support']],
'Counter-Strike: Global Offensive': ['21/08/2012', ['Valve', 'Hidden Path
      Entertainment'], ['Action', 'Free to Play'], 0, 0.0, 'Positive', 6774812, 88,
      ['win_support', 'mac_support', 'lin_support']],
'Dota 2': ['9/7/2013', ['Valve'], ['Action', 'Free to Play', 'Strategy'], 0, 0.0,
      'Positive', 1885261, 82, ['win_support', 'mac_support', 'lin_support']],
'Artifact': ['28/11/2018', ['Valve'], ['Strategy'], 1, 3.588, 'Positive', 309, 71,
      ['win_support']],
'SkiFy': ['24/01/2018', ['Blup Games'], ['Casual', 'Indie', 'Simulation'], 1,
      0.504, 'Mixed', 18, 55, ['win_support']]
}
```

```
year = 2018
```

The function will return:

```
['Artifact', 'Bloons TD 6', 'DayZ', 'SkiFy']
```

b. Parameters : `master_D (dictionary)` and `year (int)`

c. Returns : `a sorted list of games (list of strings)`

by_genre (master_D,genre) → list:

a. This function filters out games that are of a specific `genre` from the main dictionary you have created in the `read_file` function (`master_D`). It creates a list of game `names` sorted by percentage positive reviews in descending order. If there is a tie in the percentage positive reviews, keep the same order as in the dictionary. For example:

```
master_D = {
'Grand Theft Auto V': ['13/04/2015', ['Rockstar North'], ['Action', 'Adventure'],
       1, 0.0, 'Positive', 349924, 78, ['win_support']],
'DayZ': ['13/12/2018', ['Bohemia Interactive'], ['Action', 'Adventure', 'Massively
       Multiplayer'], 0, 33.588, 'Positive', 55690, 87, ['win_support']],
'Bloons TD 6': ['17/12/2018', ['Ninja Kiwi'], ['Strategy'], 1, 5.268, 'Positive',
      442, 90, ['win_support', 'mac_support']],
'Counter-Strike: Global Offensive': ['21/08/2012', ['Valve', 'Hidden Path
      Entertainment'], ['Action', 'Free to Play'], 0, 0.0, 'Positive', 6774812, 88,
      ['win_support', 'mac_support', 'lin_support']],
'Dota 2': ['9/7/2013', ['Valve'], ['Action', 'Free to Play', 'Strategy'], 0, 0.0,
      'Positive', 1885261, 82, ['win_support', 'mac_support', 'lin_support']],
'Artifact': ['28/11/2018', ['Valve'], ['Strategy'], 1, 3.588, 'Positive', 309, 71,
      ['win_support']],
'SkiFy': ['24/01/2018', ['Blup Games'], ['Casual', 'Indie', 'Simulation'], 1,
      0.504, 'Mixed', 18, 55, ['win_support']]
}
```

```
genre = 'Action'
```

The function will return:
`['Counter-Strike: Global Offensive', 'DayZ', 'Dota 2', 'Grand Theft Auto V']`
Since `'Grand Theft Auto V'` has a lower percent `positive reviews` than `'Dota 2'`, `'DayZ'` and `'Counter-Strike: Global Offensive'`.

b. Parameters : `master_D (dictionary)` and `genre (list)`
c. Returns : sorted list of game names.

`by_dev (master_D, developer) → list:`

a. This function filters out games that are made by a specific `developer` from the main dictionary you have created in the `read_file` function (`master_D`). It creates a list of game `names` sorted from latest to oldest released games. If there is a tie in the release year, keep the same order as in the dictionary. Here you would only sort by year and not have to worry about the month or day. For example:

`master_D` =
```
{
'Grand Theft Auto V': ['13/04/2015', ['Rockstar North'], ['Action', 'Adventure'],
       1, 0.0, 'Positive', 349924, 78, ['win_support']],
'DayZ': ['13/12/2018', ['Bohemia Interactive'], ['Action', 'Adventure', 'Massively
       Multiplayer'], 0, 33.588, 'Positive', 55690, 87, ['win_support']],
'Bloons TD 6': ['17/12/2018', ['Ninja Kiwi'], ['Strategy'], 1, 5.268, 'Positive',
      442, 90, ['win_support', 'mac_support']],
'Counter-Strike: Global Offensive': ['21/08/2012', ['Valve', 'Hidden Path
      Entertainment'], ['Action', 'Free to Play'], 0, 0.0, 'Positive', 6774812, 88,
      ['win_support', 'mac_support', 'lin_support']],
'Dota 2': ['9/7/2013', ['Valve'], ['Action', 'Free to Play', 'Strategy'], 0, 0.0,
      'Positive', 1885261, 82, ['win_support', 'mac_support', 'lin_support']],
'Artifact': ['28/11/2018', ['Valve'], ['Strategy'], 1, 3.588, 'Positive', 309, 71,
      ['win_support']],
'SkiFy': ['24/01/2018', ['Blup Games'], ['Casual', 'Indie', 'Simulation'], 1,
      0.504, 'Mixed', 18, 55, ['win_support']]
}
```

`developer` = `'Valve'`

The function will return:
`['Artifact', 'Dota 2', 'Counter-Strike: Global Offensive']`

b. Parameters : `master_D (dictionary)` and `developer (list)`
c. Returns : sorted list of game names

`per_discount (master_D,games,discount_D) → list:`

a. This function accepts as an argument the main dictionary you have created in the `read_file` function (`master_D`), a list of games (`games`), and the discount dictionary that you created in the `read_discount` function (`discount_D`). The function calculates and returns a list of the discounted price for each game in the list of games rounded to 6 decimal digits. If a game does not have a discount its original price is returned. We assume all games in the games list exist in `master_D`. The discounted price is calculated as $\left(1 - \frac{discount}{100}\right) * price$. For example:

`master_D` =
```
{
'Grand Theft Auto V': ['13/04/2015', ['Rockstar North'], ['Action', 'Adventure'],
       1, 0.0, 'Positive', 349924, 78, ['win_support']],
'DayZ': ['13/12/2018', ['Bohemia Interactive'], ['Action', 'Adventure', 'Massively
```

```
       Multiplayer'], 0, 33.588, 'Positive', 55690, 87, ['win_support']],
'Bloons TD 6': ['17/12/2018', ['Ninja Kiwi'], ['Strategy'], 1, 5.268, 'Positive',
      442, 90, ['win_support', 'mac_support']],
'Counter-Strike: Global Offensive': ['21/08/2012', ['Valve', 'Hidden Path
      Entertainment'], ['Action', 'Free to Play'], 0, 0.0, 'Positive', 6774812, 88,
      ['win_support', 'mac_support', 'lin_support']],
'Dota 2': ['9/7/2013', ['Valve'], ['Action', 'Free to Play', 'Strategy'], 0, 0.0,
      'Positive', 1885261, 82, ['win_support', 'mac_support', 'lin_support']],
'Artifact': ['28/11/2018', ['Valve'], ['Strategy'], 1, 3.588, 'Positive', 309, 71,
      ['win_support']],
'SkiFy': ['24/01/2018', ['Blup Games'], ['Casual', 'Indie', 'Simulation'], 1,
      0.504, 'Mixed', 18, 55, ['win_support']]
}
```

games = ['Bloons TD 6', 'Dota 2','SkiFy']

discount_D = {"Tom Clancy's Rainbow Six Siege": 60.06, "No Man's Sky": 50.03, 'SCUM': 50.07, 'X4: Foundations': 60.03, 'SkiFy': 50.0, 'X3: Reunion': 53.0}

The function will return:

```
[5.268, 0.0, 0.252]
```

b. Parameters : `master_D (dictionary)`, `games (list)` and `discount_D (dictionary)`
c. Returns : list of discounted prices

by_dev_year (master_D,discount_D,developer,year) → list:

a. This function filters out games by a specific `developer` and released in a specific `year`. The final list needs to be sorted in increasing prices. Note if a game has a discount, then its discounted price is the price that should be considered. For example,

master_D =

```
{'Grand Theft Auto V': ['13/04/2015', ['Rockstar North'], ['Action', 'Adventure'],
      1, 0.0, 'Positive', 349924, 78, ['win_support']],
'DayZ': ['13/12/2018', ['Bohemia Interactive'], ['Action', 'Adventure', 'Massively
      Multiplayer'], 0, 33.588, 'Positive', 55690, 87, ['win_support']],
'Bloons TD 6': ['17/12/2018', ['Ninja Kiwi'], ['Strategy'], 1, 5.268, 'Positive',
      442, 90, ['win_support', 'mac_support']],
'Counter-Strike: Global Offensive': ['21/08/2012', ['Valve', 'Hidden Path
      Entertainment'], ['Action', 'Free to Play'], 0, 0.0, 'Positive', 6774812, 88,
      ['win_support', 'mac_support', 'lin_support']],
'Dota 2': ['9/7/2013', ['Valve'], ['Action', 'Free to Play', 'Strategy'], 0, 0.0,
       'Positive', 1885261, 82, ['win_support', 'mac_support', 'lin_support']],
'Artifact': ['28/11/2018', ['Valve'], ['Strategy'], 1, 3.588, 'Positive', 309, 71,
      ['win_support']],
'SkiFy': ['24/01/2018', ['Blup Games'], ['Casual', 'Indie', 'Simulation'], 1,
      0.504, 'Mixed', 18, 55, ['win_support']],
'X3: Terran Conflict': ['16/10/2006', ['Egosoft'], ['Action', 'Simulation',
      'Strategy'], 1, 6.228, 'Positive', 2801, 95, ['win_support', 'mac_support',
      'lin_support']],
'X3: Reunion': ['21/07/2006', ['Egosoft'], ['Strategy'], 1, 4.428, 'Positive', 57,
      84, ['win_support', 'mac_support', 'lin_support']],
'X2: The Threat': ['21/07/2006', ['Egosoft'], ['Strategy'], 1, 3.108, 'Positive',
      2506, 86, ['win_support']]
}
```

discount_D = {"Tom Clancy's Rainbow Six Siege": 60.06, "No Man's Sky": 50.03, 'SCUM': 50.07, 'X4: Foundations': 60.03, 'SkiFy': 50.0, 'X3: Reunion': 53.0}

developer = 'Egosoft'

year = 2006

The function will return:

['X3: Reunion', 'X2: The Threat', 'X3: Terran Conflict']

'X3: Reunion' price is 369 with 53% discount, 'X2: The Threat' price is 259, and 'X3: Terran Conflict' price is 519.

b. Parameters : master_D (dictionary), discount_D (dictionary), developer (str), year (int)
c. Returns : sorted list of games

by_genre_no_disc (master_D, discount_D, genre) → list:

a. This function filters out games by a specific genre that do not offer a discount on their price. The final list needs to be sorted from cheapest to most expensive. For example:
master_D =

```
{'Grand Theft Auto V': ['13/04/2015', ['Rockstar North'], ['Action', 'Adventure'],
      1, 0.0, 'Positive', 349924, 78, ['win_support']],
'DayZ': ['13/12/2018', ['Bohemia Interactive'], ['Action', 'Adventure', 'Massively
      Multiplayer'], 0, 33.588, 'Positive', 55690, 87, ['win_support']],
'Bloons TD 6': ['17/12/2018', ['Ninja Kiwi'], ['Strategy'], 1, 5.268, 'Positive',
      442, 90, ['win_support', 'mac_support']],
'Counter-Strike: Global Offensive': ['21/08/2012', ['Valve', 'Hidden Path
      Entertainment'], ['Action', 'Free to Play'], 0, 0.0, 'Positive', 6774812, 88,
      ['win_support', 'mac_support', 'lin_support']],
'Dota 2': ['9/7/2013', ['Valve'], ['Action', 'Free to Play', 'Strategy'], 0, 0.0,
       'Positive', 1885261, 82, ['win_support', 'mac_support', 'lin_support']],
'Artifact': ['28/11/2018', ['Valve'], ['Strategy'], 1, 3.588, 'Positive', 309, 71,
      ['win_support']],
'SkiFy': ['24/01/2018', ['Blup Games'], ['Casual', 'Indie', 'Simulation'], 1,
      0.504, 'Mixed', 18, 55, ['win_support']],
'X3: Terran Conflict': ['16/10/2006', ['Egosoft'], ['Action', 'Simulation',
      'Strategy'], 1, 6.228, 'Positive', 2801, 95, ['win_support', 'mac_support',
      'lin_support']],
'X3: Reunion': ['21/07/2006', ['Egosoft'], ['Strategy'], 1, 4.428, 'Positive', 57,
      84, ['win_support', 'mac_support', 'lin_support']],
'X2: The Threat': ['21/07/2006', ['Egosoft'], ['Strategy'], 1, 3.108, 'Positive',
      2506, 86, ['win_support']]
}
```

discount_D = {"Tom Clancy's Rainbow Six Siege": 60.06, "No Man's Sky": 50.03, 'SCUM': 50.07, 'X4: Foundations': 60.03, 'SkiFy': 50.0, 'X3: Reunion': 53.0}

genre = 'Strategy'

The function will return:

['Dota 2', 'X2: The Threat', 'Artifact', 'Bloons TD 6', 'X3: Terran Conflict']

b. Parameters : `master_D (dictionary), discount_D (dictionary), genre (str)`
c. Returns : sorted list of games

`by_dev_disc (master_D, discount_D, developer) → list:`

a. This function filters out games by a specific `developer` and offers discounts. The final list needs to be sorted from cheapest to most expensive. For example,

`master_D =`

```
{'Grand Theft Auto V': ['13/04/2015', ['Rockstar North'], ['Action', 'Adventure'],
      1, 0.0, 'Positive', 349924, 78, ['win_support']],
'DayZ': ['13/12/2018', ['Bohemia Interactive'], ['Action', 'Adventure', 'Massively
      Multiplayer'], 0, 0.0, 'Positive', 55690, 87, ['win_support']],
'Bloons TD 6': ['17/12/2018', ['Ninja Kiwi'], ['Strategy'], 1, 5.268, 'Positive',
      442, 90, ['win_support', 'mac_support']],
'Counter-Strike: Global Offensive': ['21/08/2012', ['Valve', 'Hidden Path
      Entertainment'], ['Action', 'Free to Play'], 0, 0.0, 'Positive', 6774812, 88,
      ['win_support', 'mac_support', 'lin_support']],

'Dota 2': ['9/7/2013', ['Valve'], ['Action', 'Free to Play', 'Strategy'], 0, 0.0,
      'Positive', 1885261, 82, ['win_support', 'mac_support', 'lin_support']],
'Artifact': ['28/11/2018', ['Valve'], ['Strategy'], 1, 3.588, 'Positive', 309, 71,
      ['win_support']],
'SkiFy': ['24/01/2018', ['Blup Games'], ['Casual', 'Indie', 'Simulation'], 1,
      0.504, 'Mixed', 18, 55, ['win_support']],
'X3: Terran Conflict': ['16/10/2006', ['Egosoft'], ['Action', 'Simulation',
      'Strategy'], 1, 6.228, 'Positive', 2801, 95, ['win_support', 'mac_support',
      'lin_support']],
'X3: Reunion': ['21/07/2006', ['Egosoft'], ['Strategy'], 1, 4.428, 'Positive', 57,
      84, ['win_support', 'mac_support', 'lin_support']],
'X2: The Threat': ['21/07/2006', ['Egosoft'], ['Strategy'], 1, 3.108, 'Positive',
      2506, 86, ['win_support']]
}
```

`discount_D =` `{"Tom Clancy's Rainbow Six Siege": 60.06, "No Man's Sky": 50.03, 'SCUM': 50.07, 'X4: Foundations': 60.03, 'SkiFy': 50.0, 'X3: Reunion': 53.0}`

`developer` `= 'Egosoft'`

The function will return:

```
['X3: Reunion']
```

b. Parameters : `master_D (dictionary), discount_D (dictionary), developer (str)`
c. Returns : sorted list of games

`main():`

This function would read all the files and create the main dictionary and the discount dictionary. It would give users 7 different options to select from:

1. Games in a certain year
2. Games by a Developer
3. Games of a Genre
4. Games by a developer in a year
5. Games of a genre with no discount
6. Games made by a developer with discount

```
7. Exit
```

Depending on the option selected, it would prompt the user for specific inputs and display the list of sorted games as per user's selection. The main function would keep asking the user until the Exit (option 7) is selected and would display a goodbye message (check the `strings.txt` file).

1- For option 1, prompt the user to enter a `year`. If the `year` is not an `integer`, display an error message (check the `strings.txt` file). If it is a valid year and there are games that are released during that year, display all the game names sorted alphabetically separated by comma and space `', '`. Otherwise, display `"Nothing to print"`
2- For option 2, prompt the user to enter a `developer`. If there are games created by the desired developer, display all the game names sorted from latest to oldest release year and separated by comma and space `', '`. Otherwise, display `"Nothing to print"`
3- For option 3, prompt the user to enter a `genre`. If there are games that have the desired genre, display all the game names sorted by percentage positive reviews in descending order separated by comma and space `', '`. Otherwise, display `"Nothing to print"`
4- For option 4, prompt the user to enter a `developer` then a `year`. If the `year` is not an `integer`, display an error message (check the `strings.txt` file). If it is a valid year and if there are games that were made by the desired developer and are released during the desired year, display all the game names sorted in increasing prices and separated by comma and space `', '`. Otherwise, display `"Nothing to print"`
5- For option 5, prompt the user to enter a `genre`. If there are games that have the desired `genre` and do not offer a discount on their price, display all the game names sorted from cheapest to most expensive and separated by comma and space `', '`. Otherwise, display `"Nothing to print"`
6- For option 6, prompt the user to enter a `developer`. If there are games created by the desired developer and do not offer a discount, display all the game names sorted from cheapest to most expensive and separated by comma and space `', '`. Otherwise, display `"Nothing to print"`
7- For option 7, quit the program and display a goodbye message.
8- If the user enters none of these options, an invalid option message must be printed and reprompt again.

Assignment Notes and Hints

1. The coding standard for CSE 231 is posted on the course website:

   http://www.cse.msu.edu/~cse231/General/coding.standard.html

Items 1-9 of the Coding Standard will be enforced for this project.

2. The program will produce reasonable and readable output, with appropriate labels for all values displayed.

3. We provide a `proj08.py` program for you to start with.

4. If you "hard code" answers, you will receive a grade of zero for the whole project. An example of hard coding is to simply print the approximate value of e rather than calculating it and then printing the calculated average.

Assignment Deliverable

The deliverable for this assignment is the following file:

`proj08.py` – the source code for your Python program

Be sure to use the specified file name and to submit it for grading via the **Coding Rooms system** before the project deadline.

Function Tests:
We provide a zip file with all the files so it is easy to download all the files onto your computer.
Use the Function Tests if you want to test your functions locally. Make sure that your code is in the same folder as the function tests and the csv files on your computer.

I/O Test:
Look in the folder "I_O_Tests" for Tests 1-4

Grading Rubric

Computer Project #08 Scoring Summary

General Requirements:

(5 pts) Coding Standard 1-9
(descriptive comments, function headers, mnemonic identifiers, format, etc...)

Implementation:

(3 pts) open_file function (No Coding Rooms tests)

(7 pts) read_file function

(3 pts) read_discount function

(3 pts) in_year function

(4 pts) by_genre function

(5 pts) by_dev function

(5 pts) per_discount function

(4 pts) by_dev_year function

(3 pts) by_genre_no_disc function

(3 pts) by_dev_with_disc function

(3 pts) Test 1 (option 1,2,3)

(2 pts) Test 2 (option 4)

(5 pts) Test 3 (option 5 and 6)

(5 pts) Test 4 (all errors)

Note: hard coding an answer earns zero points for the whole project
-10 points for not using main()