CSC 242: Introduction to Computer Science II

FINAL EXAM PRACTICE

The final exam

The final exam will be taken on Thursday, June 11th, 2020, online. The exam begins at 9am sharp and ends at 10:30am. Submissions will be made to the D2L site, and no late submissions will be considered for any reason.

The final exam is cumulative. It can include anything covered in class, on the assignments, during the labs, and on the readings. I strongly recommend that you re-read Chapters 7 - 11 in the book and make sure that you know how to solve all the previous homework and lab problems. The material from the earlier part of the book (Chapters 1 - 6) is also assumed for the exam even though they were not explicitly assigned in this class. Assignment 10 is also concerned to be practice for the final.

There will be 2 or 3 problems. They will involve classes, GUI, recursion and web parsing. These problems may involve combining techniques (GUIS with classes, GUI with web parsing etc.).

The exam is open book. You may use up to five double-sided sheets (8 1/2 x 11 inches) or ten single-sided sheets (8 1/2 x 11 inches) of notes. The sheet(s) of notes **with your name at the top** must be handed in at the conclusion of the final exam. You may not use any other written materials. You are allowed to use the IDLE editor and the Python shell during the exam. You are not allowed to access the Internet (other than to test any functions you write) or use any other electronic devices including cell phones. You may not communicate with anyone other than Mr. Zoko during the exam. Any communication of any form with a person other than Mr. Zoko during the final exam indicates that you are done, and you will be required to immediately submit your final exam.

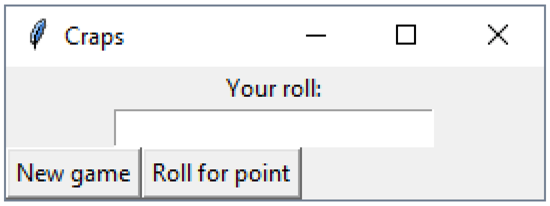
CLASSES AND GUIs

2. Implement the dice game Craps as a GUI. You must use the Dice and DiceShaker class from previous assignments. The GUI includes the button 'New Role' to simulate the initial role of the game and the Button 'Roll for point' which is clicked until the user wins or loses.

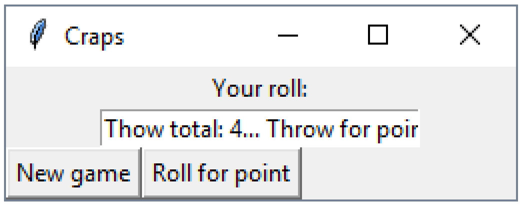
General Rules:

The 'New Role' or 'Initial Roll': If the value of shaking two dice is 7 or 11, the user wins. If the initial roll is 2,3 or 12 then the user loses. If the initial roll is anything else then the user needs to 'Roll for point'.

The 'Role for Point' the user wins if the re-roll the same point value as the initial roll. They lose if they roll a 7. They 'Roll for Point' again if the role any other value.

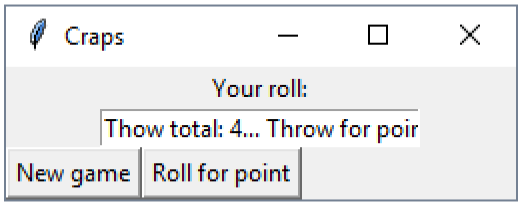


Click 'New game'



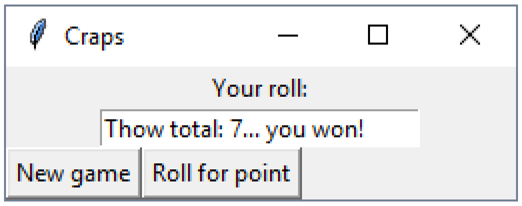
Thow total: 4... Throw for poir

Click 'Roll for point'



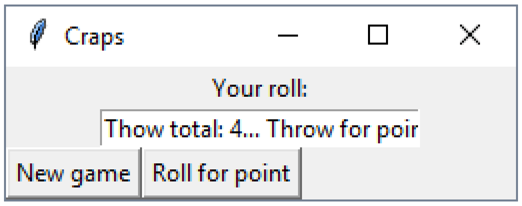
Thow total: 4... Throw for poir

Click 'New Game'



Thow total: 7... you won!

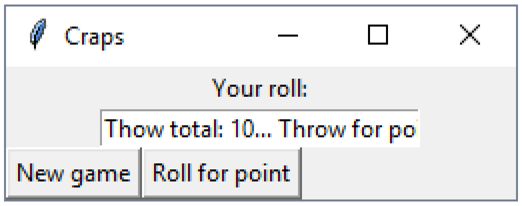
Click ‘New game’:



Your roll:

Thow total: 4... Throw for poir

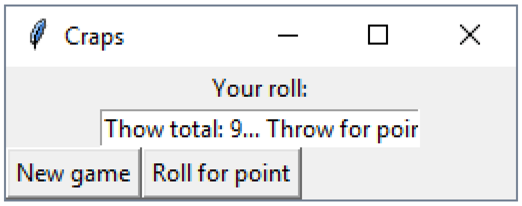
Click ‘Roll for point’



Your roll:

Thow total: 10... Throw for po

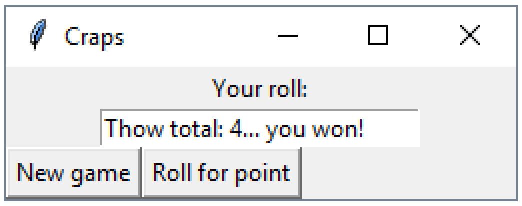
‘Click Roll for point’



Your roll:

Thow total: 9... Throw for poir

Click ‘Roll for point’

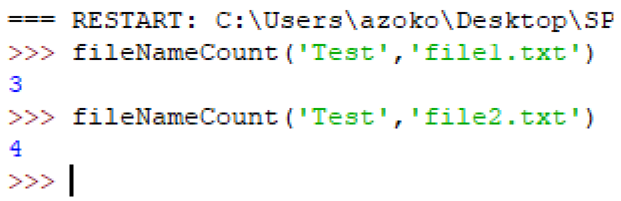


Your roll:

Thow total: 4... you won!

Recursion

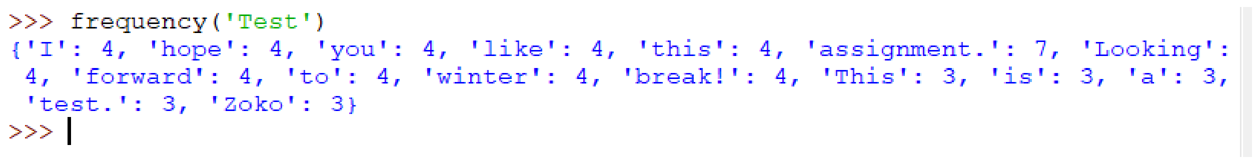
Implement a **recursive** function **fileNameCount**() that takes as parameters the name of a folder and a file name and returns a count of how many files with that name were found in the folder and subfolders. The function will return an integer representing the count of files found that match that name. Please note that your function must work as described on any directory structure, not just the one provided as an example. You must use recursion to go through the subfolders. You may not use python crawlers or apis other than the ones used in the antivirus-redux example to find files in sub directories. Doing so will be an automatic 0 for this problem. You may use a for loop to loop through the contents of a folder. The following illustrates several searches using a sample set of folders and directories located in the zip file containing the exam template. You ***don't*** need to worry about punctuation, case or file error handling for this problem.



```
=== RESTART: C:\Users\azoko\Desktop\SP
>>> fileNameCount('Test','file1.txt')
3
>>> fileNameCount('Test','file2.txt')
4
>>> |
```

Recursion

1 (25 points) Implement a **recursive** function **frequency**() that takes as parameters the name of a folder and a string to search for in the files contained within it. The function should **return** a dictionary containing the word and how many times the word was found in all files in the folder structure. Please note that your function must work as described on any directory structure, not just the one provided as an example. The following illustrates several searches using a sample set of folders and directories located in the zip file containing the assignment template. Don't worry about punctuation for this problem.



```
>>> frequency('Test')
{'I': 4, 'hope': 4, 'you': 4, 'like': 4, 'this': 4, 'assignment.': 7, 'Looking':
 4, 'forward': 4, 'to': 4, 'winter': 4, 'break!': 4, 'This': 3, 'is': 3, 'a': 3,
 'test.': 3, 'Zoko': 3}
>>> |
```

HTML Parsing

Write a HTMLParser that extracts all the courses listed in a CDM Program page. Two urls to test with can be found in the final.py file.

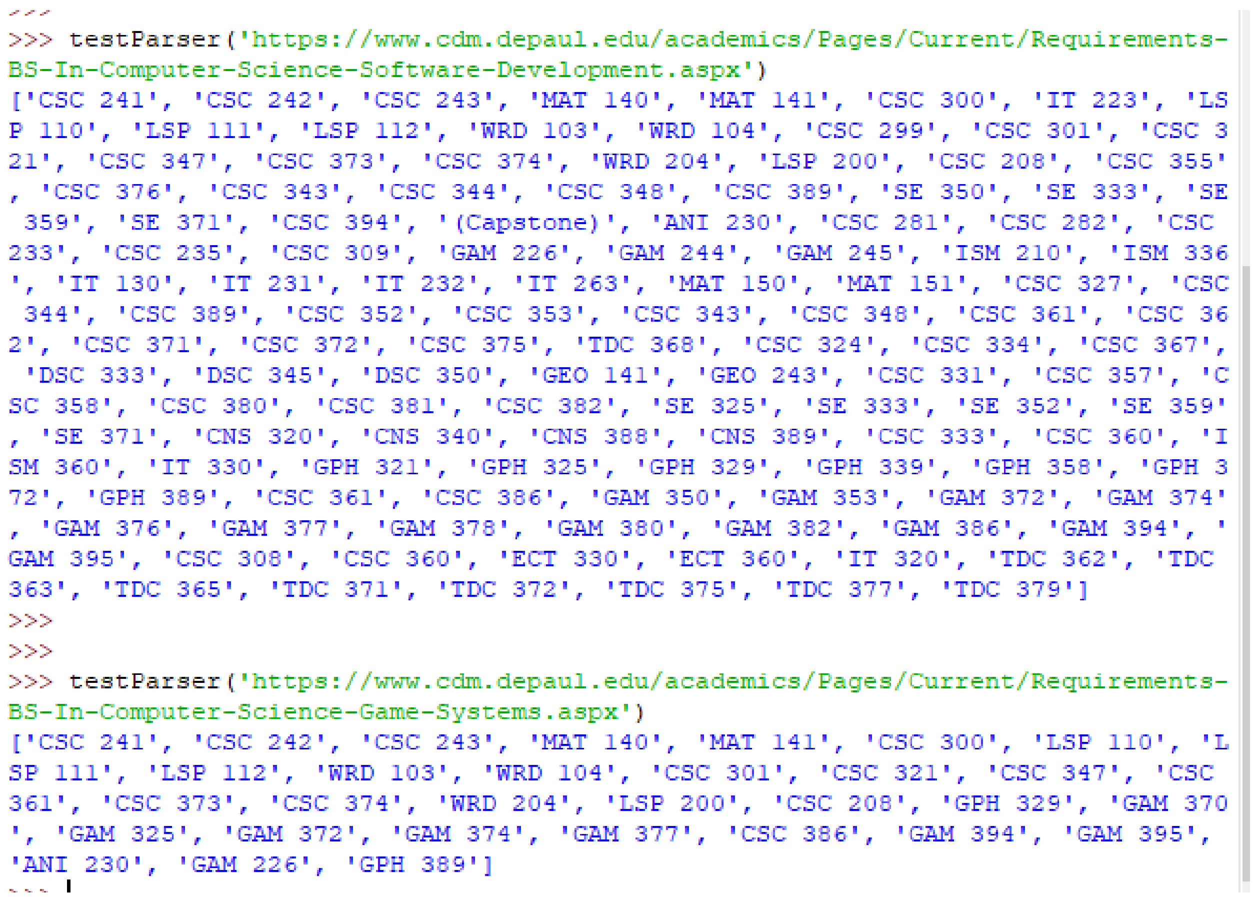
Here are examples of the html that the parser will be concerned with:

- `<span class="CDMExtendedCourseInfo">CSC 243</span>`
- `<td class="CDMExtendedCourseInfo">CSC 243</td>`

In this snippet, each course is represented in a tag with the class of “CDMExtendedCourseInfo”. The course ***can be in any tag*** but will always have a class of “CDMExtendedCourseInfo”.

You need to gather all the courses encountered and return a list.

Example:



```
>>> testParser('https://www.cdm.depaul.edu/academics/Pages/Current/Requirements-
BS-In-Computer-Science-Software-Development.aspx')
['CSC 241', 'CSC 242', 'CSC 243', 'MAT 140', 'MAT 141', 'CSC 300', 'IT 223', 'LS
P 110', 'LSP 111', 'LSP 112', 'WRD 103', 'WRD 104', 'CSC 299', 'CSC 301', 'CSC 3
21', 'CSC 347', 'CSC 373', 'CSC 374', 'WRD 204', 'LSP 200', 'CSC 208', 'CSC 355'
, 'CSC 376', 'CSC 343', 'CSC 344', 'CSC 348', 'CSC 389', 'SE 350', 'SE 333', 'SE
 359', 'SE 371', 'CSC 394', '(Capstone)', 'ANI 230', 'CSC 281', 'CSC 282', 'CSC
233', 'CSC 235', 'CSC 309', 'GAM 226', 'GAM 244', 'GAM 245', 'ISM 210', 'ISM 336
', 'IT 130', 'IT 231', 'IT 232', 'IT 263', 'MAT 150', 'MAT 151', 'CSC 327', 'CSC
 344', 'CSC 389', 'CSC 352', 'CSC 353', 'CSC 343', 'CSC 348', 'CSC 361', 'CSC 36
2', 'CSC 371', 'CSC 372', 'CSC 375', 'TDC 368', 'CSC 324', 'CSC 334', 'CSC 367',
 'DSC 333', 'DSC 345', 'DSC 350', 'GEO 141', 'GEO 243', 'CSC 331', 'CSC 357', 'C
SC 358', 'CSC 380', 'CSC 381', 'CSC 382', 'SE 325', 'SE 333', 'SE 352', 'SE 359'
, 'SE 371', 'CNS 320', 'CNS 340', 'CNS 388', 'CNS 389', 'CSC 333', 'CSC 360', 'I
SM 360', 'IT 330', 'GPH 321', 'GPH 325', 'GPH 329', 'GPH 339', 'GPH 358', 'GPH 3
72', 'GPH 389', 'CSC 361', 'CSC 386', 'GAM 350', 'GAM 353', 'GAM 372', 'GAM 374'
, 'GAM 376', 'GAM 377', 'GAM 378', 'GAM 380', 'GAM 382', 'GAM 386', 'GAM 394', '
GAM 395', 'CSC 308', 'CSC 360', 'ECT 330', 'ECT 360', 'IT 320', 'TDC 362', 'TDC
363', 'TDC 365', 'TDC 371', 'TDC 372', 'TDC 375', 'TDC 377', 'TDC 379']
>>>
>>>
>>> testParser('https://www.cdm.depaul.edu/academics/Pages/Current/Requirements-
BS-In-Computer-Science-Game-Systems.aspx')
['CSC 241', 'CSC 242', 'CSC 243', 'MAT 140', 'MAT 141', 'CSC 300', 'LSP 110', 'L
SP 111', 'LSP 112', 'WRD 103', 'WRD 104', 'CSC 301', 'CSC 321', 'CSC 347', 'CSC
361', 'CSC 373', 'CSC 374', 'WRD 204', 'LSP 200', 'CSC 208', 'GPH 329', 'GAM 370
', 'GAM 325', 'GAM 372', 'GAM 374', 'GAM 377', 'CSC 386', 'GAM 394', 'GAM 395',
'ANI 230', 'GAM 226', 'GPH 389']
```