Term	Winter 2024
Professor	Bhargava Muralidharan
Course Code	PRG600
Course Section	ZAA
Date of Final Assessment Submission	05/04/2024
Duration	3 Days (Approximately)

Instructions and Announcements

Our final assessment will be scheduled during our class time on April 5, 2024, from 4:30 pm – 5:30 pm. During this time, I will be available for assistance via email or chat. The format of this final assessment is as follows:

Mark breakdown

PART A: Multiple Choice – 10 marks

PART B: Final take home assignment – 20 marks

Total Marks – 30 marks

To be successful, please review all the topics we have covered this semester.

Please note this is a **closed book** assessment and **no aids** will be permitted.

Integrity Question / Statement

By beginning this assessment, you affirm that you will not give or receive any unauthorized help, and that all work will be your own. You agree to abide by Seneca's Academic Integrity Policy, and you understand any violation of academic integrity will be subject to the penalties outlined in the policy.

PRG600

Final Test – April 05, 2024, 11:59 PM

Worth 20 marks final grade

Submission Instructions

- Update the doc string at the top of the source code file called final.py.
- Files must be submitted as individual files NOT Zipped.
- Any screenshots of the manual test run to the required output along with any explanations must be provided in pdf file named "explanations_<studentnumber>.pdf". Ensure the final pdf has all the screenshots easy to read and clearly visible within the page.

Midterm Package

1. final.py
2. testfinal.py
3. expected_results.txt
4. winning_sample_output.txt
5. not_winning_sample_output.txt
6. Final_Exam.pdf

Required in Submission

1. final_<studentnumber>.py (Completed file)
2. actualresults_<studentnumber>.txt (This file will be created by running testfinal.py)
3. finalsubmission_<studentnumber>.txt (This file will be created by running testfinal.py)
4. flowchart_<studentnumber>.pdf
5. manual_test_output_winning.txt
6. manual_test_output_not_winning.txt
7. explanation_<studentnumber>.pdf (If you did not get 15/15 in your finalsubmission score, after running the testfinal.py and you like to provide explanations, screenshots etc use this document to submit them.(Optional)
8. Screen recording of testing the code manually (Optional)

The submission files must have student numbers in filename. If using explanations file the materials provided should go in appropriate sections. The explanations file must be saved as PDF.

The submission is very important, if the required submission guidelines are not followed no credit will be provided.

How do I attempt the test

1. Read the Test Manual
2. Create the flowchart on how you will be solving this problem.
3. Start coding and complete the functions in final.py
4. Once completed run manual tests as required using the winning_sample_output.txt and not_winning_sample_output.txt as guidance. You will have to achieve the same outputs if you were to try the same inputs, including the wordings and print statements.
5. Copy your manual test run output from the terminal into manual_test_output_winning.txt and manual_test_output_not_winning.txt respectively. (DO NOT REMOVE the ERRORS - IF YOU SEEING ANY - THIS IS IMPORTANT AS YOU COULD GET PARTIAL POINTS IN SOME CASES)
6. Run the testfinal.py script to validate your code
7. If your code is incomplete or has errors you might see some tests failed, in which case take necessary screenshots from your manual testing and the testfinal.py run output and provide them under the appropriate sections in explanations_studentnumber.docx.
8. You should also consider doing a screen recording of your manual testing and uploading that video in addition to the screenshots.
9. You should now have all required files to submit as mentioned at the beginning of this document
10. Rename the “studentnumber” in the file name into your own studentnumber and submit them (WATCH OUT FOR FILE EXTENSION). If the file name is not correct, this file will not be considered.

Interview:

The professor can call any student for a one-on-one interview to explain the code and answer some questions about the code. Failing to participate in this interview will result in a grade of zero on the final test.

General Description:

In this test, you will be writing a python code which simulates a tic-tac-toe game. If you do not know what tic-tac-toe is look at this Wikipedia page or this YouTube video.

Game Setup

Before the game starts, the game asks for the names of the players. After receiving the players’ names, the game starts with an empty 3 by 3 grid. An example of a board is shown below.

1	2	3
4	5	6
7	8	9

The numbers represent the spaces numbers that players can play.

After the grid is shown the game asks the first player to play. The first player always plays X, and the second player always plays O. The players must choose a number from the numbers that are left in the grid and press enter. If the number is not available, the game must ask the user to enter a number that has not been played before and this continues until the player selects a playable number after which the game shows the updated grid accordingly.

Playing the game

As soon as a player wins, the game congratulates the winner and prints why the winner won.

A Tie

If all the grid spaces are full but nobody has won, the game will announce that the game is a tie.

Specific Instructions

- You CANNOT use third-party libraries in your code. Python3 libraries are sufficient enough to accomplish this assignment.
- A comprehensive output along with a docstring for the functions are provided in the final.py code.
- This gives you a preliminary understanding of the functions, expected behaviour and expected output.
- winning_sample_output.txt and not_winning_sample_output.txt files have full comprehensive output expected when the final product is executed locally.

Generating output_username.txt

The final package has a winning_sample_output.txt and not_winning_sample_output..txt file the file contains the successful execution of the program for you to recreate the exact same outputs using the inputs provided.

What if my Test Script Failing

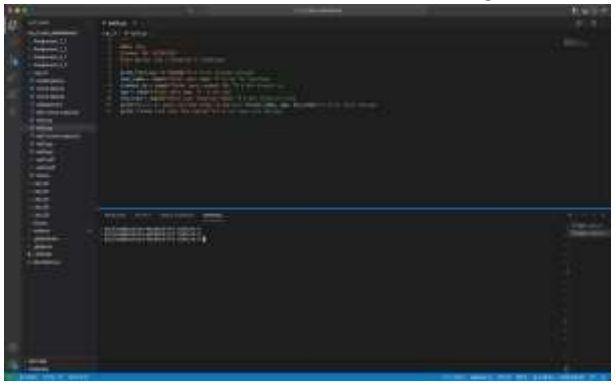
If the test scripts are failing in certain functions you need to provide screenshots and screen recordings.

- Screenshot of the error you see in the terminal when running the test script
- Successful test coverage screenshots, covering the sample input and output from the winning_sample_output.txt and not_winning_sample_output.txt. Refer to how to generate manual_test_output_winning.txt and manual_test_output_not_winning.txt for more details.
- In these cases, a screen recording will be necessary.

How to record screen

1. In Chrome Browser go to Google Screen Recorder (https://toolbox.googleapps.com/apps/screen_recorder/)

2. You do not need Audio so when an option is presented block the microphone
3. Have your VSCODE files open and the terminal opened and the command typed and ready.
4. Any unwanted applications or windows must be minimized or closed.
5. Hit the “Record” Button on the Screen Recorder Page.
6. Choose “Entire Screen”, Select Screen and hit “Share”
7. Minimize the browser and your VSCODE with the right setup should be in the foreground now.
8. Perform your testing and once done go back to the browser and hit “Stop”
9. Choose “Download” and download the recording
10. You can now submit the recording as part of your submission.



How will you be marked

When you run testfinal.py, the tester will test your code by recreating the same scenario as provided in the winning_sample_output.txt and not_winning_sample_output.txt. This tester will create a file called actualresults.txt (which you will be submitting as part of your submission).

This results is then compared against the expected_results.txt and depending on the difference you will be awarded the preliminary points.

Accuracy Percentage	Points Out of 15
100	15
95 - 100	13
90 - 95	11
85 - 90	9
80 - 85	7
75 - 80	5

70 - 75	4
60 - 70	3
50 - 60	2
20 - 50	1
0 - 20	0

Suggestion: Use the sample output files as guidance and recreate the exact same output as the sample output files using the inputs provided. Run the testfinal.py as many as you want until you achieve the desired grade for this part of the test.

Note: Ensure all necessary files and submission formats are followed. If the submission formats and guidance are not followed the preliminary score will be adjusted accordingly. Every submission will be manually reviewed, and a final score for this part of the test will be added towards the overall final grade.

Final Exam Submission

- Part 1: Available in Blackboard and must be submitted within the due date.
- Part 2:
    - A: Flowchart
    - B: Code and relevant codes submissions.

Note: You will be submitting all of Part 1 and Part 2 (A & B) between April 03, 2024, 6:00 AM to April 05, 2024, 11:59 PM.

Final Exam Grade Breakdown:

Section	**Points**
Part 1	10
Part 2 (A)	5
Part 2 (B)	15
Total	30