Cardiff School of Computer Science and Informatics

Coursework Assessment Pro-forma

Module Code:	CM1103
Module Title:	Problem Solving with Python
Lecturer:	Stuart Allen
Assessment Title:	Problem solving exercise
Date Set:	25th November 2019
Submission date and Time:	13th December 2019 at 9:30am
Return Date:	10th January 2020

This coursework is worth 40% of the total marks available for this module. If coursework is submitted late (and where there are no extenuating circumstances):

1. If the assessment is submitted no later than 24 hours after the deadline, the mark for the assessment will be capped at the minimum pass mark;
2. If the assessment is submitted more than 24 hours after the deadline, a mark of 0 will be given for the assessment.

Your submission must include the official Coursework Submission Cover sheet, which can be found here: `https://docs.cs.cf.ac.uk/downloads/coursework/Coversheet.pdf`

Submission Instructions

Description	**Type**	**Name**
Cover sheet	One pdf file	`[Student number].pdf`
Code for Q1	One Python file	`Q1_[Student number].py`
A transcript of an interactive Python session that shows the results of the test cases given for Q1	One file, either text (.txt), Python (.py)	`Q1_transcript_[Student number].py`
Answer to Q1e and figure for 1d	One Word or pdf file	`Q1_answers_[Student number].doc/docx/pdf`
Report for Q2	One Word or pdf file	`Report_[Student number].doc/docx/pdf`
Source code for Q2	One or more Python files (may be contained in a single zip file)	No restriction

Any code submitted will be run on a system equivalent to those available in the Windows/Linux laboratory and must be submitted as stipulated in the instructions above.

Any deviation from the submission instructions above (including the number and types of files submitted) may result in a mark of zero for the assessment or question part.

Assignment

This coursework asks you to compare different scoring methods for the sport of squash by simulating matches between players of different ability.

Rules of squash

This coursework uses a simplified summary of the rules of squash. Squash is a racquet game played by two players, and consists of a number of rallies. In each rally, the player who starts is the *server*, and the receiving player is the *returner*. A player wins a rally if the other player is unable to make a legal shot. *Matches* are usually played over a number of games, with each the winner being the player who reaches a certain score first. Professional matches are played over 5 games, with the winner being the first player to win 3.

There are two scoring systems commonly used in squash:

Point-a-rally scoring (PARS)

- The winner of each rally always receives a point (regardless of whether they were the server or returner).
- The first player to reach at least 11 points **and** be ahead by at least 2 points wins the game.
- If the server wins a rally, they continue as server.
- If the returner wins a rally, they become the server.

English scoring

- *Only the server is awarded a point if they win a rally.*
- If the server wins a rally, they receive a point and continue as server.
- If the returner wins a rally, they become the server but don't receive a point.
- The first player to reach 9 points wins the game **unless** the score has reached 8-8.
- If the score reaches 8-8, the player who reached 8 first decides whether to play to 9 or to 10.

Modelling playing ability

- Assume a player A's ability is represented by an *integer* value r_A such that $0 < r_A \leq 100$.
- In a game between player A and player B, the probability that A wins any given point is:

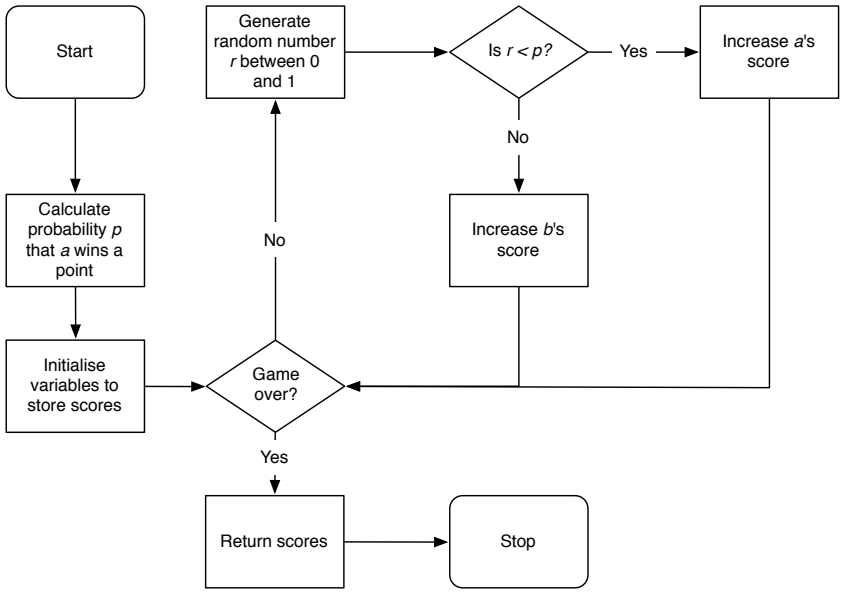
$$P(A \text{ wins}) = \frac{r_A}{r_A + r_B}$$

Questions

1. Provide code to answer the questions below. You should also provide a transcript of a Python interactive session that shows the result of executing the test case given for each question.

   All parts of this question use the *Point-a-rally scoring (PARS)* system.

   (a) Define a function `game(ra, rb)` that implements the algorithm below to simulate a single game of squash between players with abilities r_a and r_b. The function should return the final score as a tuple.



   If the random seed is set to `57`*, then calling the function with arguments* `70` *and* `30` *should return* `(11, 5)`*.*

   [Functionality: 7 marks]

   (b) Define a function `winProbability(ra, rb, n)` that simulates `n` games in order to estimate the probability that a player with ability r_a will win a game against a player of ability r_b.

   Given sufficient simulations, calling the function with abilities `70` *and* `30` *should give the answer* `0.98` *when printed to 2 decimal places.*

   [Functionality: 4 marks]

   (c) Assume players abilities are given in a csv file with the format:

```
player a ability, player b ability,
60, 20,
100, 55,
50, 40,
20, 70,
95, 85,
```

   Write a function that reads in a csv file of this format and returns a `list` of `tuples` containing each pair of abilities.

Reading in the data above should return: `[(60, 20), (100, 55), (50, 40), (20, 70), (95, 85)]`.

[Functionality: 4 marks]

(d) Write a function that takes a `list` of the format returned by your answer to 1c as an argument, and uses `matplotlib` to produce a plot (with labelled axes) showing the probability that player a beats player b (in a game) against r_a/r_b for each pair.

Show the figure produced for data `[(60, 20), (100, 55), (50, 40), (20, 70), (95, 85)]`

[Functionality: 6 marks]

(e) Suppose player a has ability 60 and player b has ability 40, and they play a match where the winner is the first player to win n games. What is the smallest value of n such that the probability that a wins the match is at least 0.9?

You may answer using simulation, theory, or a combination of both.

[Functionality: 4 marks]

2. Produce a **brief** scientific report using the supplied template (typical length should be one side of A4 – maximum allowed length is two sides of A4) that uses the results of simulation (with suitable input data) to investigate whether English or PARS is the "better" scoring method for squash matches. Your report should contain suitable figures to support your argument and clearly state any assumptions you make.

   Some starting points you may consider:

   - How *fair* is the scoring method - does the better player usually win?
   - Shorter matches are usually preferable (e.g. for television) - if you assume each rally in a match takes equal time, how long do matches typically last?
   - What effect does the relative ability of the two players have?

[Report: 7 marks; Achievement: 8 marks]

Learning Outcomes Assessed

- Use Python and common modules to implement simple algorithms expressed in pseudocode, and understand fundamental programming concepts
- Develop informal algorithms and apply recursion to solve simple problems
- Write scientific reports describing the analysis of a problem

---

Criteria for assessment

Functionality:

PASS (all marks awarded): The code displays an understanding of basic Python programming; performs the required tasks correctly for the given test data and most reasonable inputs; code may contain minor or easily corrected errors.

FAIL (no marks awarded): The code displays a lack of understanding of basic Python programming; only performs the required task for a very limited subset of inputs; contains significant errors; consists of example code with minimal changes.

Report:

6–7 marks: Relevant content covering all required elements; excellent evidence supporting conclusions; accurate; clear and concise description; clearly labelled figure.

3–5 marks: Relevant content covering many of the required elements; good evidence; few errors/omissions; generally clear; some omissions in figure.

0–2 marks: Little or no relevant content; lack of evidence; extensive errors/omissions; unclear description; no figure.

Achievement:

6–8 marks: Fully addresses scope of question.

3–5 marks: Broadly addresses scope, with some omissions or errors.

0–2 marks: Little or no achievement; many significant errors.

You will also receive feedback on the covering the quality and style of your code – this will not be included in your mark for this coursework, but is provided to help you improve your programming skills for future modules.

Code quality: Is the code elegant and well-written; easy to run; simplified by the use of built-in languages features where appropriate; readable and easy to follow? Are appropriate functions defined and written to enable reuse between questions?

Feedback

Feedback on your coursework will address the above criteria.

Individual feedback will be returned by the end of Week 12 (i.e. after 12 University working days).

Group feedback will be provided in the revision lecture and dedicated sessions in week 12, and via model solutions directly after all student submissions have been made.