ROBERT GORDON UNIVERSITY ABERDEEN TEF Gold

INFORMATICS INSTITUTE OF TECHNOLOGY

CM1601 Programming Fundamentals Assignment Specification (2020)

Module Leader	Rajitha Jayasinghe
Unit (Group/Individual)	30
Weighing	20%
Qualifying Mark	40% (both parts)
Learning Outcomes Covered in this Assignment:	LO2. Design, code, compile, test and execute fundamental programming concepts using a high-level programming language. LO3. Construct robust, maintainable programs that use object-oriented analysis and design principles
Handed Out Date	22nd October 2020
Due Date	10th December 2020
Expected Deliverable	Source code and report
Method of Submission	Online
Method of Feedback and Due Date	15th December 2020
BCS Criteria Met by this Assignment	

Assessment Regulations

Refer to the "How you are assessed section" in the Student Handbook for undergraduate students for a clarification of how you are assessed, penalties and late submissions, what constitutes plagiarism etc.

Penalty for Late Submission

Coursework received late without valid reason shall not be accepted and shall receive no grade, but shall count as one of the assessment opportunities prescribed in paragraph 9 of **RGU Academic Regulation A4 section 4.3.**

It is recognized that on occasion, illness, personal crisis or other valid circumstances can mean that you fail to submit and/or attend an assessment on time. In such cases you must inform the School of any extenuating circumstances through a ***Coursework Extension Form*** or a ***Deferral Request Form***, with valid evidence for non-submission of an assessment up to a maximum of five working days after the assessment submission date. This information will be reported to the relevant Assessment Board that will decide whether a student should be allowed to reattempt without penalty (a deferral). For more detailed information regarding University Assessment Regulations and accessing forms, please refer to the following website: www.rgu.ac.uk/academicregulations www.rgu.ac.uk/academicregulations

Grading

Marks will be awarded for the coursework based on the provided Grading Grid. These marks will be mapped onto a grade scale from A-F as determined by the individual module coordinator.

Note:

Subject to approval by external examiner

Coursework Specification

Objective

You are instructed to create a command-line program for the OMI card game in Python. Please note that this is not the traditional OMI game and to make it simple you have given a simplified version of OMI. Here, the game continues as a 2 player game including you and a robot. Consider the robot as a computer operated intelligent component. Further the computer is responsible for administrating the game.

Players and Cards

Usually, Omi is considered as a 4 player game. But to make it simple, you can consider it as 2 players including you and the robot.

A standard international 52-card pack is used, but only 32 of the cards are used for play. These cards rank from high to low A-K-Q-J-10-9-8-7 in each suit. Suits are clubs (♣), diamonds (♦), hearts (♥), and spades (♠).

Deal and Making Trumps

Here each player receives 8 cards. The computer is responsible for shuffling the cards and distributes them among two (robot and you again).

Note: Even though number of accessible cards are 32, during a one game play, only 16 cards will be used among 2 players.

After the player receives 4 cards, analyzing the available rank and suits, the player announce which suit will be trumps. After announcing trump suit, the next 4 cards will be handed over and the game starts.

The player who is responsible for announcing the trump suit should change and it should not be random too. As an example, if you declare a trump suit this time, the robot should be next.

Play

The opposite player who did not announce the trump suit, leads any card to the first trick. Then the next player, and must follow suit if able to; a player who holds no card of the suit led may play any card.

If no trumps are played, the trick is won by the highest card of the suit that was led. If any trumps were played the highest trump wins the trick. Winner of the each trick will receive +2 and lead the next trick. Whoever has the highest score at the end of 8 tricks/ whole round will be the winner of the game.

Example use cases

Imagine the trump suit is hearts (♥) and you state the suit. Now the robot can lead the first trick by playing the first card – 10 spades (10♠). Now you can check the status of your hand

- if you have J, Q, or A from ♠, you can lead the card and win the trick.

- If you have a card belongs to ♠ and that is lower than 10♠, according to the rules, you have to go ahead and eventually you lose the trick.
- However, if you do not have ♠, then you are allowed to use a trump (any card belongs to ♠) and win the trick.

Tactics

When choosing trumps, one should select the suit in which one has most cards. Between two equal suits one of which contains the ace, it is better to make the other suit trumps, since the ace is likely to win a trick whether it is a trump or not.

It is difficult to choose trump when holding cards of different suits - essentially one has to trust to luck. Some players pick the suit of the lowest card in this situation. Some players, holding four cards in three suits, with two low cards in one suit, will choose as trumps the suit in which they have no cards, hoping either to get trumps in the second part of the deal.

Tasks

1. Card shuffling algorithm (LO2)
   The programmer is instructed to add randomness to the algorithm. Both players need to have a fair hand and it should not follow a predefined pattern. But only 8x2 cards out of 32 will be selected for a given play.
   Hint: all suits and card levels can be represented as numbers and using the random module, this can be achieved.

   Shuffling has to be completed before the game starts and shuffled cards need to be saved beforehand.
   Further, cards order after the algorithm executed needs to be stored. The programmer needs to consider both level and the suit when storing
   Hint: Using 2D lists will be an added advantage.

2. Both the robot and the human player needs to follow the same tactics when deciding the trump suit. The programmer has to define those tactics as rules. The program only displays 4 cards to choose the trump suit (LO2).

   ```
   Welcome to OMI

   You are responsible to choose the trump suit

   Your hand : 10♠,J♥,8♠,7♦

   Please enter the trump suit:
   ```

3. The player who states the trump suit and then the player who leads the trick should not be the same. Next, the game phase starts. Depends on the order human user asked to enter the card. The player needs to see the available hand all the time. But the program should not mention the opposite player's hand (LO2).

   a. If the robot is leading the trick, card can be chosen randomly.

   ```
   Lets play
   ```

```
Trick 1
Trump suit : ♠
Your hand : 10♠,J♥,8♠,7♦,10♥, K♦, A♠, 8♠
Robot says : A♥
Your call : J♥
Robot wins +2

Trick 2
Trump suit : ♠
Your hand : 10♠,8♠,7♦,10♥, K♦, A♠, 8♠
Robot says : J♦
Your call :K♦
You wins +2
```

b. If the Robot following a trick, then the best card for the situation will be chosen according to the rules that have mentioned in the above description.

```
Trick 3
Trump suit : ♠
Your hand : 10♠,8♠,7♦,10♥, K♦, A♠, 8♠
Your call: ---
Robot's call :---
----- wins +2
```

4. Every time the human player disobeys the rules, the program should not execute that step and instruct user the correct procedure (LO2).

5. After 8 tricks game will be over and the program needs to announce the winner and the score. If the user wants to start another round, the program should reset the state and all the steps will be rerun. Please note that you need to write the logic in order to decide the winner of each trick (LO2).

```
Your score: 10 / Robot score:22
Robot won.
Do you want to play another round : (y/n)
```

6. Create flowcharts for the card shuffling algorithm and game phase (LO2)

7. Add a test plan to cover the winner selection for a given trick. All test cases need to have a proper description, expected, actual outputs. You do not need to add the status of the test cases as it is not required to implement the test cases (LO3).

Marking scheme

Criteria	**Marks**
Flow charts • Shuffling algorithm • Game phase	 2.5 2.5
Implementation • Shuffling algorithm • Displaying cards and trump suit selection with tactics	 5 5

• Game phase : Human + Robot's role	5
Validations • Following rules (display cards, trump suit selection, during game phase) • Error messages	 5 5
Additional improvement • Improvements **only** after the completion of mandatory requirements	 2.5
Test plan (follow the requested format)	5
Viva	12.5