Final Project

Goal: This project is to help you merge everything you have learned this semester in a few select problems. You should hand in at least one *.py* file for each of the below problems. Please name and number the problems appropriately (something like *finalProject_1_YourName.py*). Note that you will also have to include any modules developed as part of the problems.

Each code should contain extensive comments in the style discussed in class and output should be clearly displayed and labeled. Every single class should have a large block of comments describing how the class works. Similarly, the start of each problem should have a detail description of what the code does. You are welcome to comment in your own style, but you may want to look at some standard Python commenting styles (Google 'numpy style docstrings'). You *will* be graded on your programming style. Classes, functions, and lists should all be used when it makes sense. Duplicate code should be minimized.

1. (10 pts) Create a code that allows the user to play a round of blackjack against a standard computer player that hits on a 17. You can ignore any betting. You need to have at least two classes:

   – *Card* : Represents a single card. Should have at least two data elements *suit* and *rank* .
   – *Deck* : Represents a deck of cards. Should have at least one data element, a list containing 52 elements of type *Card* .

   The rest of the class/function/style is completely up to you. A few suggestions and reminders:

   – The computer gets two cards, but only shows one to the player.
   – The player hits until they decide to stand. If the player's hand value is ever above 21, they automatically lost.
   – Once the player is done (they stand), the computer draws cards until their hand value is greater than or equal to 17. If they bust, the player wins. Otherwise, you compare the player's hand value and the computer's hand value.
   – Any face card (and the ten card) are worth 10.
   – An ace is worth 11, unless the 11 makes it so the user busts, then the ace turns into a 1. For example, this hand is worth 15: ASAHAD9C3H

   Display the results of the round nicely to the screen. For example:

```
You are showing:AS 7D
The computer is showing:8D
Hit (H) or Stand (S)? S
You are showing :AS 7D
The computer is showing:TH 8D
You tied
```
   or
```
You are showing:6D QC
The computer is showing:4H
Hit (H) or Stand (S)? H
You are showing:6D QC 3C
The computer is showing:4H
Hit (H) or Stand (S)? S
You are showing :6D QC 3C
The computer is showing:9C 4H JS
The computer busted and you win!
```

2. (5 pts) Four people are playing a game of "Going to Boston". The rules are as follows:

   – Each player rolls three dice.
   – Each player keeps their highest die and sets it aside.

- The remaining two dice are re-rolled and the highest of the two is set aside.
- The last die is rolled and the final score is the sum of the three dice.

Write a code that

- Contains a class, called Player, which has at least data members: name and dice (a list of the three numbers, one for each die)
- In the Player constructor, pass in the player's name and initialize the dice to an empty list.
- The *__str__* ought to return the values of the "currently saved dice"

Your code should print out each person's score round by round. At the end of the third roll, a winner should be displayed. If any players have a tie score, each person with a high score should be declared as a winner.

3. (5 pts) Find the course average for someone in one of my mathematics class. The grading scheme is as follows:

   - 12 homework assignments, 10 points each, worth a total of 25% of the final grade. The lowest homework grade is dropped.
   - 3 regular exams, 100 points each, worth a total of 50% of the final grade
   - 1 final exam, 100 points, worth a total of 25% of the final grade
   - If a student does better on their final exam than their *lowest* regular exam grade, then replace their lowest exam grade with the grade on the final. So if a student has exam grades of 80, 74, and 52 then receives a 67 on their final, the grade of 52 will be replaced with a grade of 67.

   The grades can be found in *grades.txt*. The format is the student name on the first row, followed by each of the 12 HW grades, the next 3 exam grades, and the last a final exam grade (each grade is on its own row). This pattern then repeats for the next student. There are 5 student scores in the file. You should store all homework assignments in a list, and all regular exams in another list. Print out *just* each student's final average in the class, for example: *Paul Dostert: 87%*

4. (5 pts) Create an approximation to the US Flag, but using the following rules:

   - The stars should be drawn as sprites
   - You should not have more than 3 individual draw/blit statements (excluding the background color).
   - Draw statements can be in loops, or you can draw an entire sprite set.
   - Let the user pick from 3 sizes, small, medium or large. The height for each size should be 130, 390 and 650 pixels.

   In the comments, describe where you derived your flag ratios from (there are many different acceptable versions of the flag, so I want to know which one you used).

5. (15 pts) Create a simple video game using Pygame. You game must have

   - A winning condition
   - A losing condition
   - Some sort of score
   - Some keyboard and/or mouse controls
   - Be fun
   - Be well balanced (not too easy, not too hard)

   The game must use sprites.The game does not need to be complicated, but should be relatively unique compared to games implemented in class (do *not* turn in Pong or some variation of Pong). Watch some videos of old Atari 2600, Colecovision, Commodore 64 or Apple IIe games for some inspiration.