Problem A. Matching Cards [20 points]

Problem

Suppose two shuffled decks of cards are placed on a table, and then cards are drawn from the tops of the decks one at a time and compared. On average, how many matches do you think will occur? Write a program to carry out this process 10,000 times and calculate the average number of matches that occur. A possible output is shown in the example I/O section below (the number could be different).

Function

- matchTwoDecks() : declares two decks, shuffles them randomly, compares them and returns the number of matches.

Restrictions

- There are 52 cards in a deck of cards.

Skeleton Code

```
import random
NUMBER_OF_TRIALS = 10000


def matchTwoDecks():
    ...    # your code

    return numMatches


def main():
    totalMatches = 0

    for i in range(NUMBER_OF_TRIALS):
        totalMatches += matchTwoDecks()
    averageMatches = totalMatches / NUMBER_OF_TRIALS

    print ("Average number of matched cards: {0.3f}".format(averageMatches)


main()
```

Example I/O

```
Average number of matched cards: 1.013
```

Submit format

- HW04_A_(NAME).py

Problem B. Sequence of Numbers [20 points]

Problem

Write a recursive function that displays the sequence of numbers. A possible output is shown in the example I/O section below.

Function

– displaySequenceNumbers(m,n) : displays the sequence of numbers from m to n, where m<=n.

Restrictions

– The program must use a recursive function. i.e. displaySequenceNumbers() must call the displaySequenceNumbers() function itself inside.

Skeleton Code

```
def displaySequenceNumbers(m,n):
    ...    # your code


def main():
    print (“output of print (displaySequenceNumbers(2,4))”)
    print (displaySequenceNumbers(2,4))
    print (“output of print (displaySequenceNumbers(2,4))”)
    print (displaySequenceNumbers(3,3))


main()
```

Example I/O

```
output of print (displaySequenceNumbers(2,4))
2
3
4
output of print (displaySequenceNumbers(3,3))
3
```

Submit format

– HW04_B_(NAME).py

Problem C. Quiz Grades [20 points]

Problem

An instructor gives six quizzes with quiz grades 0 through 10, and drops the lowest grade. Write a program to find the average of the remaining five grades. A possible output is shown in the example I/O section below. The program should use a class named *Quizzes* that has an instance variable to hold a list of the six grades, a method named *average*, and a *__str__* method.

Methods

– average() : calculates the average of grades.

– __str__() : returns as shown in the example I/O (Quiz average: 9.4).

Function

– main() : declares an empty list listOfGrades, request 6 quiz grades as inputs, appends each of them to listOfGrades, and prints Quizzes(listOfGrades).

Restrictions

– Quiz grades should be taken as float.

Skeleton Code

```
class Quizzes:
    def __init__(self, listOfGrades):      # implement methods
    def average(self):
    def __str__(self):


def main():
    ...    # your code

   q = Quizzes(listOfGrades)
   print (q)

main()
```

Example I/O

```
Enter grade on quiz 1: 9
Enter grade on quiz 1: 10
Enter grade on quiz 1: 5
Enter grade on quiz 1: 8
Enter grade on quiz 1: 10
Enter grade on quiz 1: 10
Quiz average: 9.4
```

Submit format

– HW04_C_(NAME).py

Problem D. Reduce a Fraction [20 points]

Problem

Create a class named *Fraction* having instance variables for numerator and denominator, and a method that reduces a fraction to lowest terms by dividing the numerator and denominator by their greatest common divisor. A possible output is shown in the example I/O section below.

Methods

- getNumerator(), getDenominator() : accessor of each local variable.
- setNumerator(), setDenominator() : mutator of each local variable.
- GCD(): returns the greatest common divisor (GCD) of two nonzero integers. The implementation is given.
- reduce(): reduces a fraction to lowest terms by dividing the numerator and denominator by their greatest common divisor. Use GCD().

Restrictions

- numerator and denominator should be taken as int.

Skeleton Code

```
class Fraction:
    def __init__(self, numerator=0, denominator=1):  #implement methods
    def setNumerator(self, numerator)
    def getNumerator(self):
    def setDenominator(self, denominator):
    def getDenominator(self):
    def GCD(self, m, n):  #Greatest Common Divisor
            while n != 0:
                t = n
                n = m % n
                m = t
            return m
    def reduce(self):


def main():
    ...    # your code


main()
```

Example I/O

```
Enter numerator of fraction: 12
Enter denominator of fraction: 30
Reduction to lowest terms: 2/5
```

Submit format

- HW04_D_(NAME).py

Problem E. Rock, Scissors, Paper [20 points]

Problem

Write a program to play a three-game match of “rock, scissors, paper” between a person and a computer. A possible output is shown in the example I/O section below, where the last line should be changed to “TIE” in case of a tie. The program should use a class named *Contestant* having two subclasses named *Human* and *Computer*. After the person makes his or her choice, the computer should make its choice at random. The *Contestant* class should have instance variables for *name* and *score*.

Methods

- getName(), getScore() : accessor of each local variable.
- incrementScore(): adds 1 point to score.
- makeChoice(): Human requests a choice from rock, scissors, paper as input (repeats the request until the input is valid) and returns the choice. Computer randomly makes the choice and returns it.

Functions

- main(): Requests Human and Computer’s name as inputs, and create Human and Computer objects with the names. Performs matches using playGames() and shows the final result. (Display “TIE” if tie)
- The implementation of the function playGames() and higher() is given.

Restrictions

- The data types of name and score are string and int, respectively.
- Display the final result in uppercase.

Skeleton Code

```
class Contestant:
    def __init__(self, name=””, score=0):  #implement methods
    def getName(self):
    def getScore(self):
    def incrementScore(self):

class Human(Contestant):
    def makeChoice(self):

class Computer(Contestant):
    def makeChoice(self):

def playGames(h, c):
   for i in range(3):
       choiceH = h.makeChoice()
       choiceC = c.makeChoice()
       if choiceH == choiceC:
           pass
       elif higher(choiceH, choiceC):
           h.incrementScore()
       else:
           c.incrementScore()
       print(h.getName() + ":", h.getScore(),
             c.getName() + ":", c.getScore())
       print()

def higher(c1, c2):
   if ((c1 == 'rock' and c2 == 'scissors') or
       (c1 == 'paper' and c2 == 'rock') or
       (c1 == 'scissors' and c2 == 'paper')):
       return True
   else:
       return False

def main():
   ...     # your code

main()
```

Example I/O

```
Enter name of human: Garry
Enter name of computer: Big Blue

Garry, enter your choice: rock
Big Blue chooses paper
Garry: 0 Big Blue: 1

Garry, enter your choice: scissors
Big Blue chooses rock
Garry: 0 Big Blue: 2

Garry, enter your choice: paper
Big Blue chooses scissors
Garry: 0 Big Blue: 3

BIG BLUE WINS
```

Submit format

– HW04_E_(NAME).py