CS507 Computing Foundation for Computational Science HW5

Shell Programming

Min Long

Instruction

Due time: Due: 11/18/2022, 23:59
Submission command: `submit minlong CS507 HW5`

Instruction to electronic submission: http://cs.boisestate.edu/∼cs221/SubmissionProcedure.html

- The written assignment can be done in a pure text format (*.txt, for example, problem1.txt) on Onyx.
- The programming assignment should be presented with source codes.
- Each problem should have its own working directory, such as HW2/prob1, HW2/prob2 ... For example, the following table shows the structure of HW1 from the user "student1" and how to submit HW to us through Onyx

```
[student1@onyx:HW1]$ ls -l
drwxr-x---. 2 student1 Students  13 Sep 19  2021 prob1
drwxr-xr-x. 3 student1 Students  14 Sep 19  2021 prob2
[student1@onyx:HW1]$ cd prob1/
[minlong@onyx:prob1]$ ls
-rw-r-----. 1 student1 Students 943 Sep 19  2021 problem1.txt
$ cd ..
[student1@onyx:HW1]$ pwd
/home/student1/CS507/HW1
[student1@onyx:HW1]$ submit minlong CS507 HW1
```

Listing 1: A sample structure of homework and submission procedure.

- Your source codes (if any) must compile and run on Onyx.
- Documentation is important and proper comments are expected in your source code.
  - comments giving description of: purpose, parameters, and return value if applicable
  - other comments where clarification of source code is needed
  - proper and consistent indentation
  - proper structure and modularity

Don't ask us or your classmates directly for solutions (it happened); just try as much as possible. Be patient and enjoy coding!

Programming Problems

The grading will be based on the functionality, quality, simplicity, and user-friendliness of the code. The simpler, better-designed, shorter code will get a higher score compared to barely runnable code.

1. (10 pts) **Class and Object**. Please help a busy professor to make a code **using a class** to compute students' letter grades based on the average of his/her mid-term and final exams. The mapping between the score and the grade can be found below.

Average Score	≥ 90	≥ 80	≥ 70	≥ 60	< 60
Letter Grade	A	B	C	D	F

   The first program is called lgStudent.py. It should include at test main() and a LGstudent class. In the class, we expect a constructor and the following functions setName(), setMidterm(), setFinal(), calcGrade() and __str__() functions.

   The second program is the client code called grade.py, which should import the LGstudent class from lgStudent.py.

   The program will read input from the terminal and provide output like:

```
$ python grade.py
Enter student's name: Amy
Enter student's grade on midterm exam: 90
Enter student's grade on final exam: 92
Do you want to continue (Y/N)? Y
Enter student's name: Fred
Enter student's grade on midterm exam: 84
Enter student's grade on final exam: 97
Do you want to continue (Y/N)? N

NAME   GRADE
Amy      A
Fred     A
```

Listing 2: Sample output.

2. (10 pts) **Class and Object**. Create a class named Fraction, having instance variables for numerator and denominator. It also should have the following method: constructor, setNumerator(), getNumerator(), setDenominator(), getDenominator(), __str__(), a method GCD() that computes the greatest common divisor reduces a fraction to lowest terms by dividing the numerator and denominator by their greatest common divisor. Save the class in the file fraction.py.

   (a) (5pts) Reduce a Fraction. Write a program that import the Fraction class and requests a fraction as input and reduces the fraction to lowest terms.

```
$ python fraction-reduce.py
Enter numerator of fraction: 12
Enter denominator of fraction: 30
Reduction to lowest terms: 2/5
```

Listing 3: Sample output.

   (b) (5pts) Convert a Decimal to a Fraction Write a program that converts a decimal number to an equivalent fraction.

```
$ python fraction-convert.py
Enter a positive decimal number less than 1: .75
Converted to fraction: 3/4
```

Listing 4: Sample output.

3. (10pts)) **Class and Object**. Proceed to Checkout Write a program that checks out the items in the user's cart on a shopping website. The program should use a class named Purchase to hold the information about a single item purchased (that is, description, price, and quantity) and a class named Cart to hold a list whose items are objects of type Purchase.

   The Purchase class should include constructor, setDescription(), getDescription(), setPrice(), getPrice(), setQuantity(), getQuantity() functions.

   The Cart class should include constructor, addItemToCart(), getItems() and calculateTotal() functions.

```
$ python purchase-client.py
Enter description of article: apple
Enter price of article: 2
Enter quantity of article: 3
Do you want to enter more articles (Y/N)? Y
Enter description of article: banana
Enter price of article: 1
Enter quantity of article: 10
Do you want to enter more articles (Y/N)? N

ARTICLE      PRICE QUANTITY
apple        $2.00     3
banana       $1.00    10

TOTAL COST: $16.00
```

Listing 5: Sample output.

4. (10pts) **Inheritance**. Let's extend the code obtained from the Problem 1.

   Create two subclasses LGstudent(Student) and PFstudent(Student) that inherit all the properties and methods of a superclass Student. LGstudent class handles students letter grades. PFstudent class handles whether students Pass or Fail a class.

   Obviously, the PFstudent class will have a calcGrade() method different from that in the LGstudent class as shown below.

Average Score	≥ 60	< 60
Pass or Fail	Pass	Faill

   Next write a drive code student-client which imports student and accomplish the following jobs.

```
$ python student-client.py
Enter student's name: Bob
Enter student's grade on midterm exam: 70
Enter student's grade on final exam: 85
Enter category (LG or PF): LG
Do you want to continue (Y/N)? Y
Enter student's name: Amy
Enter student's grade on midterm exam: 92
Enter student's grade on final exam: 83
Enter category (LG or PF): LG
Do you want to continue (Y/N)? Y
Enter student's name: Fred
Enter student's grade on midterm exam: 75
Enter student's grade on final exam: 76
Enter category (LG or PF): PF
Do you want to continue (Y/N)? N

NAME  GRADE
Amy B
Bob C
Fred  Pass
Number of letter-grade students: 2
Number of pass-fail students: 1
```

Listing 6: Sample output.

5. (10pts) **Inheritance**. Rock, Paper, Scissors Write a program to play a three-game match of “rock, paper, scissors” between a person and a computer. See Fig. 7.15. 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. (Note: Rock beats scissors, scissors beats paper, and paper beats rock.)

```
$ python game.py
Enter name of human: Amy
Enter name of computer: DeepMind

Amy, enter your choice: rock
DeepMind chooses scissors
Amy: 1  DeepMind: 0

Amy, enter your choice: scissors
DeepMind chooses paper
Amy: 2  DeepMind: 0

Amy, enter your choice: paper
DeepMind chooses paper
Amy: 2  DeepMind: 0

AMY WINS
```

Listing 7: Sample output.