CSC015 Assignment 6 (100 pts)

What to Submit: The codes for each problem (as separate .py files)

The purpose of this assignment is for you to top-down program design with functions..

Naming conventions: follow the usual naming conventions., hw#_problem#_yourname.py.

Follow style rules. Include a program comment and a header comment of each function, use inline comments as appropriate.

Leave an empty line between function definitions to make code easier to read.

Put the definition of main() first or last, and the call to main() last.

Problem 1. (20 points) Write a program processing the golden ratio constant $\boldsymbol{\phi}$. The value of the golden ratio constant is computed by the formula

$$\phi = \frac{1 + \sqrt{5}}{2}$$

The program

1. Prints information about the golden ratio constant
2. Prints the formula that defines how the constant is computed
3. Prints the value of the golden ration constant

For each of these steps you will use functions. Fore each step 1-3:

1. Print a message why the golden ratio constant is relevant (research this on the internet and prepare a brief 3-5 line statement that you would print on he screen for the user to see it). Implement this step as a function goldenRatioInfo(), the function does not take a parameter and does not return a value, it just print the information about the constant. You will call this function in main() to accomplish step 1.
2. Print the formula for computing the golden ratio constant as it would appears in a Python code. Implement this part as a function goldenRatioFormula(), no parameters and no return. You will call this function in main() to accomplish step 2.
3. Print the value of the golden ratio constant. Implement a function, goldenRatio(), the function does not have a parameter, it computes the value of $\boldsymbol{\phi}$ using the formula and returns the computed value. The function does not print. In main() call the function to get the value and print the value (of course state that what you are printing is the value of the golden ratio constant).

The three functions you need to write for this problem are independent and do not call each other, so you may implement them in any order you want. The calls in main must be in the order to accomplish steps 1,2,3.

During lab, in class, implement the function goldenRatio() and test it with main(), calling the function in main() and then printing the value returned. For submission do the whole problem.

Below are the function headers,

def goldenRatioInfo(): that prints the message about the constant
def goldenRatioFormula(): that prints the formula
def goldenRatio(): that computes and returns the value of the constant
def main(): puts together the program

Problem 2: (30 points) For a real number x, $x \neq 0$, and an integer number k , x^k can be computed as follows :

- If $k > 0$, $x^k = x \cdot x \cdot \cdots \cdot x$, where x is used k times in multiplications.
- $x^0 = 1$
- If $k < 0$, $x^k = \frac{1}{x^{-k}}$, where x raised to the positive power is computed as given above.

Write a function **intpower(x, k)** that takes as parameters a real number x and an integer value k and computes x^k using the above process and returns the computed value, The function does not read input form keyboard, it does not print.

You must not use module math functions for computing powers or ** (for computing x^k). You must implement the computation from scratch (use conditionals and loops as needed to implement the process given above).

The function header is

def power(x, k):

To test, write a program, with main(), the program prints a table of the integer powers of 10 from 10^{-5} *to* 10^5. Sample output is below. Do not write other functions, use just main() and intpower(). Sample output is below.

```
 k     10^k
----- ----------
 -5   1e-05
 -4   0.0001
 -3   0.001
 -2   0.01
 -1   0.1
  0   1
  1   10
  2   100
  3   1000
  4   10000
  5   100000
```

Problem 3. (50 points) Write a program to compute student grades. The program will: read a grade scale and student IDs and test scores for each student, assign grades to each student and print the average test score, the grade scale and corresponding grade , the a table of student IDs and assigned grades.

I advise that you implement incrementally and test the functions described below , one by one.

I have broken the whole program for you into steps and have specified functions that yo must use to accomplish each step.

main() will put the program together. What you need to accomplish is

1. Read grade scale from keyboard. Use a function **getScale().** The function has no parameters, it reads from the keyboard 4 integer values between 0 and 100, the values must be input in strictly increasing order. Do input validation (each entry must be valid and greater than the previous). The values are stored in a list scale[], the list scale is returned. The idea is that grades will be assigned later using the list elements as boundaries

TABLE 1

Score	Grade
score <= scale[0]	‘F’
scale[0] < score <= scale[1]	‘D’
scale[1] < score <= scale[2]	‘C’
scale[2] < score <= scale[3]	‘B’
scale[3] < score	‘A’

1. Read student IDs and test scores. Write and use a function **getData().** The function has no parameters it reads an ID (between 100 and 999) and a test score (between 0 and 100) , separated by space, from the keyboard (like 999 87, means ID is 999 and 87 is score), till sentinel empty string is entered. The values entered are stored into a dictionary scores{}, where the key is the IDs and the value is the associated score. The function returns the dictionary. Do input validation. If any of the two values entered is invalid, prompt again for input.
2. For each student compute the grade based on the test score**. Use a loop in main() and iterate over the dictionary scores{}**, for each student compute the grade. To compute the grade write and use a function **setGrade(scale, a_score)** that takes as parameters the list scale[] and the student score for one student, a_score, and computes and returns the grade. **Store the student ID and grade as a record ID:grade in a dictionary grades{}.** This dictionary is created and filled in main().
3. Print the average test score. To accomplish this use a function **average(scores)** that takes as a parameter the dictionary scores{} and computes and returns the average score. The function does not print. Main() will call the function to compute the average score and main will print the result with an appropriate message.
4. Print the grade scale, write and use a function **printScale(scale)** that takes as a parameter the grade scale and prints information for the user each line specifies the range for a grade and gives the grade , for example

   ```
   A           score > 90
   B   89 >= score > 80
   C   79 >= score > 70
   ….
   ```

   Note that I used specific values, like 90, 80 etc, you must print the values from scale[].

   The function does not return.

5. Write a function **printGrades(grades)** that takes as a parameter the dictionary grades and prints a table with 2 columns, ID and Grade . Each row is an id and grade per student. The function does not return. Main() will call this function to print the output., the grades for all students.

In summary, you will write functions

Write and use the following functions:

- **getScale( )** to read scale for computing grades and returns a list. Do input validation the list must be of integers between 0 and 100 in strictly increasing order.
- **getData()** reads strings formed by IDs and test scores (in this order, space separated) until empty string entered, input is validated, 10<= ID <= 99, 0<=score<=100. Repeat

prompt till valid value is entered. The function fills a dictionary scores{} as discussed above and returns it.

- **setGrade(scale, a_score)** computes and returns a grade per student given grade scale, scale[], and student score, a_score.
- **average(scores)** given as parameter the dictionary scores{} it computes and returns the average test score (a real number)
- **printScale(scale)** given as parameter the grade scale, scale[], and prints information about the grade assignment based on scores and scale (as described above)

  A score > 90
  B 89 >= score > 80
  C 79 >= score > 70
  ….
  Use the scale[] not the numbers I used in the example.
- **printGrades(grades)** takes as a parameter the dictionary grades{} and prints a table with first column the ID and second grade for all records in grades.