Homework Assignment 2

General information:

Responsible teaching assistant: Moshe Davidian.
Submission date: 28.4.22 at 23:50

Students' Questions regarding the assignment will be answered only in the HW2 forum.

If necessary, office hours are on Thursday, 13:00-14:00. Please send an email to coordinate prior.

Pay Attention:

- Do not use external libraries to solve this exercise. (No import statements allowed)
- Some tests will be visible for your convenience, and others will not.
- The assignment must be done individually. Similarity tests will be performed automatically, and similar codes will be automatically graded as 0.
- Template code with the main block code is given. Do not modify the template!
- Write the code only under the "#WRITE YOUR CODE HERE." comment.
- Good luck!

Question 1

Implement the function **q1** that reads values from an input file and returns the result of dividing of the numbers in the order in which they appear.

- The input file is located in the code folder and can contain several lines.
- The function gets the file name as parameter (**input_file_name**).
- It is necessary to take care of cases of division by 0 and reading non-number values, in which case a code must be returned according to the following details:

Error	Return value
A problem in opening the file	-1
Division by 0	-2
Reading non-number values	-3

For example:

Input_q1.txt	Return value
4096 2 4 8 2 8	4.0
4096 2 0 8 2 8	-2
4096 a 4 8 0 8	-3
461214 3 4.5 *5 6.5* *7.3 8* *2*	9.0

Question 2

Implement the function **q2** that reads words from an input file and writes the words into a new output file sorted in ascending order by the length of the word (as elements in a list as shown below).

- The input file is located in the code folder and can contain several lines.
- The function gets the file name as parameter (**input_file_name**).
- The name of the output file will be "output_" + **input_file_name**.
- Words of the same length will be written in the order in which they appear in the original input file (Stable sorting).
- For each word, the word itself and the its length should be written as a tuple, for example, ('word', 4).

For example:

Input_q2.txt:

Python is a high-level, general-purpose programming language.
Its design philosophy emphasizes code readability with the use of significant indentation.

Output_q2.txt:

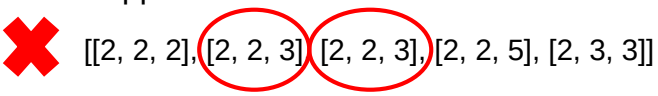
[('a', 1), ('is', 2), ('of', 2), ('Its', 3), ('the', 3), ('use', 3), ('code', 4), ('with', 4), ('Python', 6), ('design', 6), ('language.', 9), ('philosophy', 10), ('emphasizes', 10), ('high-level,', 11), ('programming', 11), ('readability', 11), ('significant', 11), ('indentation.', 12), ('general-purpose', 15)]

Question 3

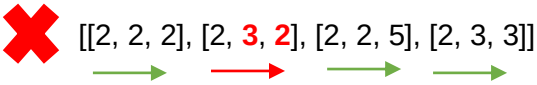
A. Implement the function **q3_a** that receives an integer number **n** and returns a list of all prime numbers up to **n**. The numbers in the list will be arranged in ascending order.

B. Implement the function **q3_b** that receives an integer number **n** and returns a list that contains all the three prime numbers combinations whose product is less than or equal to **n**. The function **q3_b** needs to use function **q3_a** (that you wrote in section A).

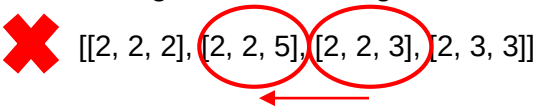
- Each combination appears once in the returned list:



- Each combination is arranged in ascending order:



- All combinations are arranged in ascending order:



<u>For example:</u>

for n = 20, the function returns:

```
print(q3_b(20))
```

```
[[2, 2, 2], [2, 2, 3], [2, 2, 5], [2, 3, 3]]
```

for n = 30, the function returns:

```
[[2, 2, 2], [2, 2, 3], [2, 2, 5], [2, 2, 7], [2, 3, 3], [2, 3, 5], [3, 3, 3]]
```

Question 4

Implement the function **q4** that receives a number **n** and returns the n-th line in the Pascal triangle as in the example.

- **n** can be assumed to be positive.
- **n** not necessarily integer (in this case, the function returns the code -1).

Explanation of Pascal Triangle: Wikipedia

For example:

n	Return
`print(q4(0))`	`[1]`
`print(q4(2))`	`[1, 2, 1]`
`print(q4(6))`	`[1, 6, 15, 20, 15, 6, 1]`
`print(q4(2.5))`	`-1`

Question 5

A. Implement the function **q5_a** that receives 4 arguments as input: **z**, **a**, **b**, and **n**. **z** represents a function $z(x)$. The function returns the approximation of the root of $z(x)$ ($z(x) = 0$) according to the bisection method. Arguments **a** and **b** are numbers that represents the starting interval $[a, b]$. The argument **n** represents the number of iterations in the bisection method. If $z(a_i)z(b_i) \geq 0$ in the i-th iteration of the method when $1 \leq i \leq n$, and $[a_i,\ b_i]$ represents the interval in this iteration, the function returns the Boolean value False.

The accuracy of the final answer will be rounded to 2 digits after the dot.

For example:

parameters	Return value

`def g(x):`
`  return 2*x+5`
`print(q5_a(g, -10, 10, 10))`
	`-2.5`

`def g(x):`
`  return x**2+5`
`print(q5_a(g, -10, 10, 10))`
	`False`

`def g(x):`
`  return x**2-5`
`print(q5_a(g, -10, 10, 10))`
	`False`

`def g(x):`
`  return x**2-5`
`print(q5_a(g, 0, 15, 10))`
	`2.23`

`def f(x):`
`  return 3*x+9`
`print(q5_a(f, -10, 10, 10))`
	`-3.0`

B. Implement the function **q5_b** that receives 5 arguments as input: **f**, **g**, **a**, **b**, and **n**. **f** and **g** represent functions $f(x)$ and $g(x)$, respectively. The function returns the approximation of a solution $f(x) = g(x)$ (using the bisection method) where **a**, **b**, and **n** has the same functionality as in 5A. Hint: Use the function you wrote in the previous section.

The accuracy of the final answer will be rounded to 2 digits after the dot.

For example:

parameters	Return value

`def f(x):`
`  return 2*x+9`
 	
`def g(x):`
`  return 2*x+5`
`print(q5_b(f,g, -10, 10, 10))`
	`False`

`def f(x):`
`  return 2*x+5`
 	
`def g(x):`
`  return -2*x+5`
`print(q5_b(f,g, -10, 10, 10))`
	`0.0`

`def f(x):`
`  return 2*x+10`
 	
`def g(x):`
`  return 3*x+5`
`print(q5_b(f,g, -100, 100, 10))`
	`4.98`

`def f(x):`
`  return 2*x+10`
 	
`def g(x):`
`  return 3*x+5`
`print(q5_b(f,g, -10, 2, 10))`
	`False`