CS 1301 – Homework 7

Recursion
Due: Friday November 4th, 11:55pm
File to submit to T-Square: HW7.py

You should work individually on this assignment. You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. Students may only collaborate with fellow students currently taking CS 1301, the TA's and the lecturer. You should not exchange code or write code for others. For individual assignments, each student must turn in a unique program. Your submission must not be substantially similar to another student's submission. **Collaboration at a reasonable level will not result in substantially similar code.**

For Help:

- TA Helpdesk – Schedule posted on class website.
- Email TA's or use Piazza Forums

Notes:

- **Do not forget to include the required collaboration statement** (outlined on the syllabus).
- **Do not wait until last minute** to do this assignment in case you run into problems.
- **Read the entire specifications document before starting this assignment.**

Recursion

Function Name: `square_contents` (20 points)

Description: Define a recursive function that accepts a list of numbers as a parameter and returns a new list with all the initial contents raised to the power of 2. You may not use any looping in your solution.

Your solution must use recursion in order to receive points.

Parameter(s):

1. A list containing integers or floats.

Return Value: A list containing squared integers and floats.

Example(s):

```
>>> print(square_contents([1, 2, 3]))
[1, 4, 9]
>>> print(square_contents([1, 2, 3.5]))
[1, 4, 12.25]
```

Function Name: `factorial_dictionary` (30 points)

Description: Define a recursive function accepts a number as a parameter and returns a dictionary. The dictionary should contain, as a key, every number from 1 to the given number. The corresponding value for each key should be the factorial of that number. You may not use any looping in your solution.

Your solution must use recursion in order to receive points.

Parameter(s):

1. A single positive integer.

Return Value: A dictionary containing each integer and it's corresponding factorial.

Example(s):

```
>>> print(factorial_dictionary(1))
{1: 1}
>>> print(factorial_dictionary(5))
{1: 1, 2: 2, 3: 6, 4: 24, 5: 120}
```

Function Name: `solve_eq` (50 points)

Description: Define a recursive function that finds the result of an arithmetic string. The given string will contain numbers, operators (+, -, *, /), and possibly spaces. You should compute the result respecting order of operations. The function should return an int if the string contains only whole numbers and does not use division. Otherwise, the function should return a float. You do not need to consider division by zero in your solution.

Your solution must use recursion in order to receive points.

Parameter(s):

1. A string containing numbers, operators (+, -, *, /), and possibly spaces.

Return Value: An int or a float of the resulting value.

Example(s):

```
>>> print(solve_eq("1 + 1"))
2
>>> print(solve_eq("1 + 5 * 4"))
21
>>> print(solve_eq("1.0 + 5 * 4"))
21.0
>>> print(solve_eq("1 - 10 * 5 / 80"))
0.375
```

Rubric

`square_contents`	**20**
contains a base case	5
contains a recursive call	5
returns correct result using recursion	10
`factorial_dictionary`	**30**
contains a base case	5
contains a recursive call	5
returns correct result using recursion	20
`solve_eq` *(must be recursive to receive points)*	**50**
returns the correct value when order of operations is irrelevant	15
returns the correct value (regardless of type) in all cases	30
result type is always correct	5