Ontario Tech University

Faculty of Science

Course:	CSCI 1030U: Introduction to Computer Science
Description:	Practice programming problems #2 (For Test #2 Preparation)

Question 1

Write a Python function (`reverse_rec`) that takes a single string and returns the reverse of that string using recursion.

Note: Do this question without using `[::-1]` or `reverse()`.

Function name	**Description**
`range(x, y, z)`	Returns a list of numbers between `x` and `y` (not including `y`), spaced apart by `z`. Example: `range(0, 7, 2) => [0,2,4,6]`
`len(s)`	Returns the number of elements in the list `s`. Example: `len([0,2,4,6]) => 4`

`str[i]`	Returns the character at index `i` in `str`.
	
	Example:
	`str = "Natasha"`
	`str[4] => "s"`

`str1 + str2`	Returns the concatenation of `str1` and `str2`.
	
	Example:
	`str1 = "Gabe"`
	`str2 = "Newell"`
	`str1 + str2 => "GabeNewell"`

Permitted Functions for this question

Example output:
```
>>> reverse_rec('CSCI 1030U')
'U0301 ICSC'
>>> reverse_rec('Python!')
'!nohtyP'
```

Question 2

Write a *recursive* function (`palindrome_rec`) that takes a string (`str`), and returns `True` or `False`, depending on whether or not `str` is a palindrome.

Note: *The sample code below does not need to be copied into your answer. You are responsible for the function described above, and nothing else.*

Note: *A palindrome is a word that is spelled the same backwards and forwards, such as 'eve' or 'ablewasiereisawelba'*

Hint: *To check if a string is a palindrome, you could check the first and last characters. If they are not equal, the string is not a palindrome. If they are, the string is a palindrome if the remaining characters between the first and last character are also a palindrome. For example, within the palindrome 'level' is the palindrome 'eve'.*

Function name	**Description**
`range(x, y, z)`	Returns a list of numbers between `x` and `y` (not including `y`), spaced apart by `z`. Example: `range(0, 7, 2) => [0,2,4,6]`
`len(s)`	Returns the number of elements in the string `s`. Example: `len([0,2,4,6]) => 4`

Permitted Functions for this question

Sample Output:
```
print("is level a palindrone?",palindrome_rec("level"))
# True

print("is lever a palindrone?",palindrome_rec("lever"))
# False
```

Question 3

Write a class (`Product`) that has three instance variables:

- name (a string)
- price (a floating point number)
- weight (a floating point number)

Your class should also have four member functions:

1. a constructor – This function will store the values of the three instance variables, which will be passed as arguments in the order given above (see the sample output, below, for an example)
2. `get_shipping_cost()` – This function will calculate the cost of shipping the item using the formula:
    - `shipping_cost = weight * 10`
3. `get_tax()` – This function will calculate the tax on the item (not including shipping):
    - `tax = price * 0.13`
4. `get_total_cost()` – This function will calculate the total cost of the item:
    - `total_cost = price + tax + shipping_cost`

Note: *For this class, you can use any built-in functions that you'd like.*

Sample output:
```
razor = Product("Electric Razor", 49.99, 0.25)
home_gym = Product("Home Gym", 879.99, 115.0)

print(f'Total cost of {razor.name}: {razor.get_total_cost():0.2f}')
# Total cost of Electric Razor: 58.99

print(f'Total cost of {home_gym.name}: {home_gym.get_total_cost():0.2f}')
# Total cost of Home Gym: 2144.39
```

Question 4

Write a Python class (`Student_Entry`) that represents a student in our class. The following instance variables should be represented:

- `labs`: The mark for the student's labs (out of 10)
- `assignments`: The mark for the student's assignments (out of 20)
- `midterm`: The student's midterm mark (out of 100)
- `final`: The student's final exam mark (out of 100)
- `name`: The student's full name
- `sid`: The student's ID

Your class should include these methods:

1. A constructor: Given the name and student ID, initializes all of the variables (the marks should be zero initially)
2. `get_average():` Uses the values of the given marks to calculate the student's average. The assignment and lab marks are out of their correct weight, but the midterm is worth 30% and the final worth 40%, so those grades need to be scaled down accordingly.
3. `letter_grade():` Returns the student's grade (A,B,C,D,F), according to the following rules:
    a. A: 80-100
    b. B: 70-79
    c. C: 60-69
    d. D: 50-59
    e. F: 0-49
4. A comparison function allowing two `Student_Entry` instances to be compared using the < (less than) operator

Note: *For this class, you can use any built-in functions that you'd like.*

Example output:
```
bsmith = Student_Entry("Bob Smith", "1000001")
bsmith.labs = 9.0
bsmith.assignments = 17.0
bsmith.midterm = 57.5
bsmith.final = 60.0
print(f'Bob Smith: {bsmith.letter_grade()}')
# output: Bob Smith: C

sjones = Student_Entry("Sally Jones", "1000002")
sjones.labs = 9.5
sjones.assignments = 19.5
sjones.midterm = 81.0
sjones.final = 74.5
print(f'Sally Jones: {sjones.letter_grade()}')
# output: Sally Jones: A
```

Question 5

Write a class (`Dictionary`) that implements a dictionary (also known as an associative array) using the list of tuples underlying data structure discussed in the lectures. This class will support the following methods:

- An initializer (initializes the list of tuples, initially empty)
- `set` (given `key` and `value` arguments, `key` being a string value, inserts a new key/value pair into the list)
- `get` (given a `key` string argument, will return the corresponding value in the dictionary)
- A string converter function (outputs the list of tuples)

Write another class (`KeyNotFoundError`) that you can use for an exception, which will be thrown when the `get` function is called with a key that isn't anywhere in the dictionary.

Note: *For this class, you must implement the class using a list of tuples. An implementation that uses a dictionary will not be accepted.*

Sample output:

```
products = Dictionary()
products.set('RTX3060', 329.99)
products.set('RTX3070', 499.99)
products.set('RTX3080', 1499.99)
products.set('RTX3090', 1999.99)

print(f'products = {products}')
# output: products = [('RTX3060', 329.99), ('RTX3070', 499.99),
#                     ('RTX3080', 1499.99), ('RTX3090', 1999.99)]

print(f'RTX3090 = {products.get("RTX3090")}')
# output: RTX3090 = 1999.99

try:
    print(f'RTX3050 = {products.get("RTX3050")}')
except KeyNotFoundError as error:
    print(f'Cannot find key: {error}')
# output: Cannot find key: Key not found: RTX3050
```

Question 6

Write a function, `math1`, which calculates a specified number of terms from the following MacLaurin series (like we did in section 03a):

$$e^{-x} = 1 - \mathrm{x} + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \cdots = \sum_{n=0}^{\infty} \left(\frac{(-1)^n \, x^n}{n!} \right)$$

The arguments of the function will be:

- `x` - the value to be used for x shown in the MacLaurin series
- `n` – the starting `n` value for which a term should be calculated
- `max_n` - the final `n` value for which a term should be calculated

Sample calls of this function and their output are given below, but your code should work for all inputs:

```
print(f'{math1(0.0, n = 0, max_n = 10) = }')
# output: math1(0.0, n = 0, max_n = 10) = 1.0

print(f'{math1(0.5, n = 0, max_n = 10) = }')
# output: math1(0.5, n = 0, max_n = 10) = 0.606530659724375
```

Question 7

Write a function, `math2`, which calculates a specified number of terms from the following MacLaurin series (like we did in section 03a):

$$erf(x) = \frac{2}{\sqrt{\pi}}(\mathrm{x} - \frac{x^3}{3\cdot 1!} + \frac{x^5}{5\cdot 2!} - \frac{x^7}{7\cdot 3!} + \cdots) = \sum_{n=0}^{\infty}\left(\frac{2(-1)^n\, x^{2n+1}}{\sqrt{\pi}\cdot(2n+1)\cdot n!}\right)$$

The arguments of the function will be:

- `x` - the value to be used for x shown in the MacLaurin series
- `n` – the starting `n` value for which a term should be calculated
- `max_n` - the final `n` value for which a term should be calculated

Sample calls of this function and their output are given below, but your code should work for all inputs:

```
print(f'{math2(0.0, n = 0, max_n = 10) = }')
# output: math2(0.0, n = 0, max_n = 10) = 0.0

print(f'{math1(0.5, n = 0, max_n = 10) = }')
# output: math1(0.5, n = 0, max_n = 10) = 0.5204998778130467
```