MERKLE

Junior Developer - Coding Test

Attempt to answer all of the coding problems below in a language you are comfortable with. Wherever possible try to avoid using imported libraries and built-in methods to handle calculations (unless absolutely necessary). If using JavaScript, stick to vanilla JS as much as possible

There is no set time limit for the test, but we anticipate it taking 2-3 hours. Please answer each question in a separate file. When complete, please host your answers in a publicly available repo (e.g. Github) and send us the link.

1. Longest Sequence

Implement the function `longestSequence(sequence)` which takes a string of letters and finds the longest sequence where the same letter occurs continuously.

The function must return an object literal {c: n} where c is the lowercase character and n is the length of this sub-sequence.

If there are multiple characters with a continuous sequence of the same length, return the information for the letter which occurs first alphabetically.

Example outputs:

```
longest_sequence( "dghhhmhmx" ) === {h: 3}
longest_sequence( "dhkkhhKKKt" ) === {k: 3}
longest_sequence( "aBbBadddadd" ) === {b: 3}
```

2. Savings Account Balance

Implement the following function to calculate the current balance of a bank savings account at the end of `numMonths` (a specific number of months).

```
balance(openingSum, interestRate, taxFreeLimit, taxRate, numMonths)
```

The calculation must reflect the following assumptions:

- `openingSum` is the initial account balance.
- Every month, the monthly interest is calculated as `interestRate` percent of the account balance available at the beginning of each month.
- Every month, the monthly tax is calculated as `taxRate` percent of (account balance - `taxFreeLimit`) if the account balance is greater than the allowed `taxFreeLimit`. Otherwise, the monthly tax is 0.

- At the end of each month, the balance is updated by adding the monthly interest and subtracting the monthly tax.

Example outputs:

```
balance(10000, 1, 20000, 1, 2) #must be approximately 10201
balance(25000, 2, 20000, 1, 2) #must be approximately 25904.5
balance(19800, 2, 20000, 1, 2) #must be approximately 20597.96
```

3. Recursive Reverse String

Implement a function `reverseString(string)` which accepts a string parameter, reverses it and returns it.

Do not use .reverse() or any other standard JavaScript string functions or sorting functions. Do not use loops or iteration. Use the following ideas to implement the function recursively:

- if the string is empty, its reverse is the empty string
- if the string has one character, return it
- otherwise, the result is a concatenation of the string parameter without the first letter, in reverse, and the first letter at the end

Example outputs:

```
reverseString( "abcb" ) === "bcba"
reverseString( "test" ) === "tset"
reverseString( "racecar" ) === "racecar"
```

4. Time Class

Implement a class called `Time`. The following class variables should be initialised as integers in the class constructor (or __init__ method in Python):

```
hours
minutes
seconds
```

Include a method which normalises the time and invoke it appropriately so that the time is automatically adjusted when large integers are passed in. The following rules should remain true:

```
0 <= hours <= 23
0 <= minutes <= 59
0 <= seconds <= 59
```

Include a method called `scale` which will accept an integer of seconds and add it to the time held in the current instance.

Include a method called `timeString` which returns a string in the format "`hh:mm:ss`".
Example outputs:

```
Time t = new Time( 1, 30, 20 );
t.timeString() == "01:30:20"
t.scale( 400 );
t.timeString() == "01:37:00"

Time t = new Time( 1, 100, 0 );
t.timeString() == "02:40:00"
```