exam_50min

Fundamentals of Computer Science - programming exam

Instructions

IMPORTANT - BEFORE YOU START:

1. Edit the `.py` file in Spyder or VS-Code and solve the exercises.
2. Upload the `.py` file to BBoard.

Additional notes

- The exam is **not open book**!
- **Clean your mess** after you finish! Comments and tests/asserts are welcome, but unnecessary code should be deleted.
- **Tests are provided** for each exercise, although they are initially commented out. They are very helpful to understand the intended behavior and to check the correctness of your implementation. Make sure to uncomment and run them as you go through the exercises.
- **WARNING**: in the following exercises, your are **not allowed** to use lists' methods such as append, insert, ..., slices or comprehensions. You are also **not allowed** to use set, dictionaries, or call functions from any library. Essentially, you will have to solve the exercises using plain conditionals and loops.

Exercise 1

Define a function called `total_occurrences` that takes as input two lists, `l1` and `l2`, and counts the number of times any element of `l1` appears in `l2`.

The elements of `l1` can be assumed to be all distinct. An exception should be raised if any of the two inputs is not a list.

The following assertions should pass:

```python
assert total_occurences([1, 2], [1, 2, 2, 3]) == 3
assert total_occurences([1, 2], [1, 2, 2, 3, 1]) == 4
assert total_occurences([1], [4, 1, 1]) == 2
```

Exercise 2

You have to define a function called `riemann` that takes as input a function `f`, two float numbers `a` and `b`, and an integer `n`. The function will provide an approximation of the integral of `f` between `a` and `b` using the Riemann sum with `n` subintervals.

The Riemann approximation is given by the right hand side of the following equation:

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} f(x_i) * \Delta$$

where Δ is the length of each subinterval, defined as $\Delta = \frac{b-a}{n}$, and $x_i = a + i * \Delta$.

The `riemann` function should check that `b` is strictly larger than `a` and raise an exception otherwise. The argument `n` should take default value `1000`.

Exercise 3

Write a function called listsplit that takes a list `l` of numbers as its only argument. The function must return an index `i` between `0` (included) and `len(l)` (excluded), such that the difference between the sum of the first `i` elements of the list and the sum of the remaining elements is the smallest possible (in absolute value).

For example, let's say that `l = [2, 4, -1, 5, -2]` . Then, we can look at this table where for each value of `i` there are the left and right half-lists, their sum, and the difference (the best split is shown in red):

```
input: l = [2, 4, -1, 5, -2]
```

i	left	right	sum left	sum right	diff
0	[]	[2, 4, -1, 5, -2]	0	8	8
1	[2]	[4, -1, 5, -2]	2	6	4
2	[2, 4]	[-1, 5, -2]	6	2	4
3	[2, 4, -1]	[5, -2]	5	3	2
4	[2, 4, -1, 5]	[-2]	10	-2	12
5	[2, 4, -1, 5, -2]	[]	8	0	8

In this case the function should return `3` . In case there are multiple indices that give the same result, return the lowest one. This last rule implies that in fact you only need to get up to `len(l)-1` , as stated at the beginning. In fact, the case `i==len(l)` is the last line of the table above, which clearly cannot be better than the first anyway, so we don’t even need to consider it.

You don’t need to check the arguments of the function for this exercise.

IMPORTANT: for this exercise, you cannot use slicing expressions such as `l[0:i]` , nor the built-in function `sum` . You can (and should) use the built-in function `abs` though. Also, for the full score, you need to solve this without nested loops (i.e. no loops within other loops...). Using nested loops is still considered a very good solution though.