Advanced Python: Homework set 3

2023/2024

Each task is worth 4 points. Two items must be presented for evaluation. Where it makes sense, use the list deque implementation from the collections module, which implements adding and removing elements from the ends of a list more efficiently than standard Python lists.

Problem 1

Program a function that implements the algorithm that finds the longest palindrome in the given text given as an argument to the function. The result is a list of tuples of the form (i, l), where i is the position of the beginning of the palindrome and l is its length. If there are no palindromes in the text, the function returns an empty list. We assume that

1. the palindrome should have a length of at least 2;
2. it is not necessary to remove punctuation marks, spaces or unify case of letters as it was in the task on the first list;
3. the algorithm should have a complexity of at most $O(n^2)$. Those interested can look for Manacher's algorithm with complexity O(n)

Problem 2

Program two functions needed to operate on arithmetic expressions in reverse Polish notation:

1. the `convert(infix_expression)` function, which returns `infix_expression` in the form of RPN. We assume that `infix_expression` is a list of numbers, brackets and operators, e.g.

```
['(', 2, '+', 3, ')', '*', 4]
```

   and the result is also a list;

2. the `calculate(rpn_expression)` function, which calculates an expression in the form RPN, where the `rpn_expression` argument is the same as the previous result functions.

Problem 3

Write an algorithm to find a way in a maze. We assume that there is a maze given as a list of lists, where obstacles are marked with an `'X'` and the absence of an obstacle space. The solution should be in the form of a function whose arguments are a maze and the coordinates of the starting point, and the result is a list of coordinate pairs places creating a path to the exit.

Problem 4

Program a list sorting algorithm that will correctly sort a list of numbers written in words, e.g

```
['one hundred and twenty-three', 'eight hundred and fifteen',\
'thirty thousand two hundred']
```

The example gives English numerals, but it is not possible to program for other languages. It is enough if the program works for six-digit numbers.

Problem 5

Program the function `max_sublist_sum(list)`, which will return list for a list of numbers such a pair `(i, j)` that the sum of `sums(list[i:j+1])` is the largest for all pairs$0 \leq i \leq j < len(list)$. Note that there may be negative numbers in the list, so solution

```
i = 0
j = len(list)—1
```

is not always the best.