Zestaw 3

1. Zaimplementuj klasę stos wykorzystując listy w języku Python, ale wstawiając element na początku listy, a nie na koniec, jak to było pokazane na wykładzie. Jakie są asymptotyczne czasy operacji na stosie przy takiej implementacji?

2. Ciągiem nawiasowym (zbudowanym tylko z nawiasów ()[]{} <> (bez spacji!) nazywamy dowolną permutację nawiasów otwierających i zamykających. Ciąg nawiasowy jest poprawny, gdy można go uzupełnić liczbami i działaniami tak, aby utworzone wyrażenie matematyczne było wykonalne. Przykładowo ciąg {{[()()]}} jest poprawny, lecz ciąg ([])()] nie jest poprawny. Należy sprawdzić, czy podane na wejściu ciągi nawiasowe są poprawne.

   Napisz stosowną funkcję, która na wejściu dostaje ciąg nawiasowy (maksymalnie 1000 znaków), a zwraca słowo TAK, jeśli ciąg jest poprawny, lub słowo NIE w przeciwnym przypadku.

3. Mamy dane wyrażenie, składające się z n nawiasów otwierających i n zamykających (używamy teraz tylko nawiasów klasycznych postaci ()). Tym razem na wejściu zakładamy, że jest to poprawne wyrażenie nawiasowe (patrz zadanie wcześniej). Jan napisał bardzo długie wyrażenie nawiasowe, a ukończywszy swe dzieło, ze zgrozą zauważył, że sam już nie wie, które nawiasy zamykające odpowiadają, którym otwierającym. Pomóż Janowi i wyznacz pary nawiasów, które sobie odpowiadają. Napisz stosowną funkcję, która na wejściu dostaje poprawny ciąg nawiasowy złożony z $2n$ nawiasów (maksymalnie 2000000 znaków). Wyjście powinno składać sią z n wierszy, z których i-ty wiersz zawiera dwie liczby całkowite a_i oraz b_i, gdzie $1 \leqslant a_i < b_i \leqslant 2n$), oznaczające, że nawias otwierający na a_i-tej pozycji odpowiada nawiasowi zamykającemu na b_i-tej pozycji (zakładamy, że pozycje te numerujemy od 1). Pary te powinny być wypisane w kolejności rosnących numerów pozycji nawiasów zamykających, czyli $b_1 < b_2 < ... < b_n$. Jak będzie wyglądać wyjście dla układu ()((()(()))?

4. Zapisz wyrażenia $(2 + 4) * 6 - 8$ i $((2 + 7)/3 + (14 - 3) * 4)/2$ w notacji postfiksowej (odwrotnej notacji polskiej). Następnie pokaż, że wynik obliczenia wartości wyrażenia w ONP jest zgodny z wynikiem dla notacji infiksowej.

5. Napisz funkcję w języku Python, który w oparciu o stos oblicza wartość wyrażenia zapisanego w notacji postfiksowej. Zakładamy, że korzystamy jedynie z 4 klasycznych operacji arytmetycznych: dodawania, odejmowania, mnożenia i dzielenia. Przykładowo dla danych wejściowych postaci $20\ 10\ +\ 75\ 45\ -\ *$ wynik powinien wynosić 900. Funkcję proszę nazwać Postfix Eval().