**Лабораторная работа №4**

**Полный перебор**

Выполнил: Морошкин Рудольф, 9 вариант

Задание:

Скобочные выражения

Будем считать, что скобочное выражение состоит только из круглых

открывающихся и закрывающихся скобок.

Примеры:

(())()

()()

Скобочное выражение может быть правильным (см. примеры выше) и

неправильным, например,

)(()

(()))

())(

Написать программу, выводящую все правильные скобочные выражения

данной длины N (N<=20) и подсчитывающей их количество. При N>6 результаты

выводить в текстовый файл.

Код:

class Stack:

def \_\_init\_\_(self):

self.\_values = []

def push(self, value):

self.\_values.append(value)

def pop(self):

if self.is\_empty:

return None

return self.\_values.pop()

def peek(self):

if not self.is\_empty:

return self.\_values[0]

else:

return None

@property

def is\_empty(self):

return len(self.\_values) == 0

def match\_braces(braces: int, length: int, stack: Stack) -> bool:

total\_braces\_dont\_have\_pairs = 0

for brace\_n in range(1, length+1):

bit = 1 << (length - brace\_n)

brace = bool(braces & bit) # getting a brace at position

if brace == 1:

total\_braces\_dont\_have\_pairs += 1

if brace == 0:

stack.push(brace)

elif brace == 1 and stack.peek() == 0:

total\_braces\_dont\_have\_pairs -= 1

stack.pop()

return stack.is\_empty and total\_braces\_dont\_have\_pairs == 0

def is\_valid\_braces\_expression(braces: int, length: int) -> bool:

# if we got odd number of braces - they can't match

if length % 2: return False

# Starting with ) brace - we can't match

if braces & (1 << (length-1)): return False

return match\_braces(braces, length, Stack())

def encode\_to\_braces(number: int, length: int):

"""

Returns a string from braces, where 0 bit = (, and 1 bit = ).

:param number: a number that we got to transform into braces.

:param length: length of expected braces string.

:returns: string of braces.

"""

braces = "(", ")"

braces\_out: str = ""

for bit\_n in range(length):

bit\_number = number & (1 << bit\_n)

braces\_out = braces[bool(bit\_number)] + braces\_out

return braces\_out

def generate\_braces():

for length in range(1, 21):

for number in range(0, 2\*\*length):

if is\_valid\_braces\_expression(number, length):

yield encode\_to\_braces(number, length)

total\_matching\_braces = 0

with open("output.txt", mode="w") as out:

for n, braces\_text in enumerate(generate\_braces()):

if len(braces\_text) > 6:

out.write(f"{braces\_text}\n")

else:

print(braces\_text)

total\_matching\_braces += 1

print("Total matching braces:", total\_matching\_braces)

Тест:

