Algorytmy i struktury danych
Laboratorium 4

mgr inż. Bartłomiej Kizielewicz, mgr inż. Andrii Shekhovtsov

30 listopada 2023

1 Zasady oceniania

Ocena za laboratorium zależy od liczby dobrze zrobionych zadań domowych, skala ocen jest pokazana w Tabeli 1. W niektórych przypadkach mogą być uwzględnione zadania rozwiązane częściowo.

Tabela 1: Skala ocen.

Liczba zrobionych zadań domowych	Ocena
1	3.0
2	4.0
3	5.0

Wykonane zadania proszę przesłać za pośrednictwem platformy moodle w postaci pliku tekstowego z kodem w Python (plik z rozszerzeniem **.py**). Proszę wrzucić kod ze wszystkich zadań do jednego pliku, rozdzielając poszczególne zadania za pomocą komentarzy.

UWAGA: Termin oddania zadania jest ustawiony w systemie moodle. W przypadku nie oddania zadania w terminie, uzyskana ocena będzie zmniejszana o 0,5 za każdy zaczęty tydzień opóźnienia. Zadania oddawane później niż miesiąc po terminie ustawionym na moodle są oddawane i rozliczane w trybie indywidualnym na zajęciach lub po umówieniu się z prowadzącym.

UWAGA: W przypadku wysłania zadania w formie niezgodnej z opisem w instrukcji prowadzący zastrzega prawo do wystawienia oceny negatywnej za taką pracę. Przykład: wysłanie `.zip` lub `.pdf` tam, gdzie był wymagany plik tekstowy z rozszerzeniem `.py`.

2 Pogotowie Pythonowe

Przykład używania stosu w Python:

```
>>> from collections import deque
>>> stos = deque()  # Tworzymy stos
>>> stos.append(42)  # Dodajemy element do stosu
>>> stos  # Zawartość stosu
deque([42])
>>> stos.append(3)  # Dodajemy kolejny element na górę stosu
>>> stos
deque([42, 3])
>>> stos.pop()  # Zdejmujemy ostatnio dodany do stosu element
3
>>> stos
deque([42])
```

Przykład używania kolejki w Python:

```
>>> from collections import deque
>>> kolejka = deque()  # Tworzymy kolejke (tak, to ta sama klasa)
>>> kolejka.append(42)  # Dodajemy element do końca kolejki
>>> kolejka  # Zawartość kolejki
deque([42])
>>> kolejka.append(3)  # Dodajemy kolejny element na koniec kolejki
```

```
>>> kolejka
deque([42, 3])
>>> kolejka.popleft()  # Zdejmujemy element z początku kolejki
42
>>> kolejka
deque([3])
```

3 Zadania do wykonania na zajęciach

1. Napisz funkcję, sprawdzającą czy wszystkie otwarte nawiasy okrągłe w napisie zostały zamknięte prawidłowo. Funkcja ma wykorzystywać do tego celu stos.

   Przykład:

   ```
   >>> isvalid('(())')
   True
   >>> isvalid('((2 + 5) * (2 + 3)) / 2')
   True
   >>> isvalid('(()')
   False
   >>> isvalid('))((')
   False
   ```

2. Napisz funkcję która obliczy wartość wyrażenia matematycznego zapisanego w postfiksowej postaci (odwrotna notacja polska). Do obliczeń należy wykorzystać stos.

   ```
   >>> calculate('2 3 + 5 *')
   25
   >>> calculate('2 7 + 3 / 14 3 - 4 * + 2 /')
   23.5
   ```

3. Napisać funkcję symulującą działanie oprogramowania „planisty" w systemie operacyjnym. W rzeczywistych systemach operacyjnych program ten przydziela poszczególnym procesom w systemie zasoby procesora tak, by dla użytkownika wyglądało to tak, jakby wszystkie procesy działały naraz.

   Implementowana funkcja powinna przyjmować listę krotek, gdzie każda krotka określa nazwę procesu oraz liczbę milisekund potrzebnych do zakończenia procesu, a także ile milisekund pracy procesora powinno być przydzielano (zmienna `n`). Lista procesów powinna być następnie potraktowana jako kolejka procesów do wykonania, z którą należy postępować następująco:

   (a) Pobierz proces z kolejki.

   (b) Zmniejsz czas który pozostał mu do zakończenia o `n`.

   (c) Jeżeli proces się zakończył (w wyniku dostaliśmy czas 0 lub poniżej), to zapisujemy jego nazwę na listę procesów zakończonych.

   (d) Jeżeli procesowi jeszcze pozostał jakiś czas pracy, to z prawdopodobieństwem `p` dodaj do pozostałego czasu wykonania losową liczbę z zakresu $[0, 1000)$. I dopisz ten proces na koniec kolejki.

   Jako wynik działania funkcji proszę zwrócić listę zakończonych procesów w takiej kolejności w jakiej one były kończone, a także nazwy procesów którym zabrakło czasu (nie wykonały się w czasie `max_time`) - będą to procesy pozostałe w kolejce gdy skończy nam się czas.

   ```
   # Przykładowe uruchomienie programu i przykładowe wyniki.
   processes = [
       ('git status', 250),
       ('python calculations.py', 5340),
       ('gcc main.c', 13400),
       ('screenshot', 1500)
   ```

```
]
print(scheduler(processes, n=100, p=0.2, max_time=5*60*1000))
# (['git status', 'screenshot', 'python calculations.py'], ['gcc main.c'])
```

Podpowiedź: Warunek działający z pewnym prawdopodobieństwem można napisać w taki sposób:

```
prob = 0.2
if random.random() < prob:
    print(f'Ten wiersz wyświetli się z prawdopodobieństwem {prob}.')
else:
    print(f'Ten wiersz wyświetli się z prawdopodobieństwem {1 - prob}.')
```

4 Zadania do wykonania w domu

1. Napisz funkcję, sprawdzającą czy wszystkie otwarte nawiasy w napisie zostały zamknięte prawidłowo. Funkcja ma wykorzystywać do tego celu stos i obsługiwać dwa typy nawiasów: okrągłe i kwadratowe.

   Przykład:

```
>>> isvalid('(())')
True
>>> isvalid('([])')
True
>>> isvalid('((2 + 5) * (2 + 3)) / 2')
True
>>> isvalid('a = [(3, 5), (2, 5), (2, 9)]')
True
>>> isvalid('](')[]')
False
>>> isvalid('[[([]]))')
False
```

2. Napisać funkcje, symulującą działanie placówki „Poczta Polska". Funkcja ta ma otrzymać na wejściu listę krotek, zawierających po kolei imię klienta oraz zmienną True/False która oznacza czy klient będzie musiał wrócić do okienka po raz drugi (by np. wypełnić potwierdzenie nadania i wrócić wysłać list). Na końcu działania funkcja powinna zwrócić listę osób wychodzących z poczty w tej kolejności w której one będą wychodzić. Funkcja powinna używać kolejki.

   Kolejka powinna być obsługiwana następująco:

   (a) Pobierz osobę na początku kolejki.
   (b) Sprawdź czy osoba musi coś wysłać.
   (c) Jeżeli tak to dopisz na koniec kolejki.
   (d) Jeżeli nie to dopisz do listy osób wychodzących z poczty.

```
# Przykładowe uruchomienie programu i przykładowe wyniki.
line = [
    ('Grażyna', True),
    ('Laura', False),
    ('Bartek', False),
    ('Andrzej', True),
    ('Wiesiek', False)
]
print(pocztapolska(line))
# ['Laura', 'Bartek', 'Wiesiek', 'Grażyna', 'Andrzej']
```

3. Zaimplementuj algorytm porządkowania stosu za pomocą stosu pomocniczego zgodnie z pseudokodem przedstawionym na wykładzie. Algorytm także jest opisany na stronie https://eduinf.waw.pl/inf/alg/001_search/0103.php.