Algorytmy i Struktury Danych
Zadanie offline 1 (6.III.2023)

Format rozwiązań

Rozwiązanie zadania musi się składać z **krótkiego** opisu algorytmu (wraz z uzasadnieniem poprawności) oraz jego implementacji. Zarówno opis algorytmu jak i implementacja powinny się znajdować w tym samym pliku Pythona (rozszerzenie `.py`). Opis powinien być na początku pliku w formie komentarza (w pierwszej linii w komentarzu powinno być imię i nazwisko studenta). Opis nie musi być długi—wystarczy kilka zdań, jasno opisujących ideę algorytmu. Implementacja musi być zgodna z szablonem kodu źródłowego dostarczonym wraz z zadaniem. Niedopuszczalne jest w szczególności:

1. korzystanie z wbudowanych funkcji sortujących,
2. korzystanie z zaawansowanych struktur danych (np. słowników czy zbiorów),
3. zmienianie nazwy funkcji implementującej algorytm, listy jej argumentów, lub nazwy pliku z rozwiązaniem,
4. modyfikowanie testów dostarczonych wraz z szablonem,
5. wypisywanie na ekranie jakichkolwiek napisów innych niż wypisywane przez dostarczony kod (ew. napisy dodane na potrzeby diagnozowania błędów należy usunąć przed wysłaniem zadania).

Dopuszczalne jest natomiast:

1. korzystanie z następujących elementarnych struktur danych: krotka,
2. korzystanie ze struktur danych dostarczonych razem z zadaniem (jeśli takie są).

Wszystkie inne algorytmy lub struktury danych wymagają implementacji przez studenta. Dopuszczalne jest oczywiście implementowanie dodatkowych funkcji pomocniczych w pliku z szablonem rozwiązania.

Zadania niezgodne z powyższymi ograniczeniami otrzymają ocenę 0 punktów. Rozwiązania w innych formatach (np. `.PDF`, `.DOC`, `.PNG`, `.JPG`) z definicji nie będą sprawdzane i otrzymają ocenę 0 punktów, nawet jeśli będą poprawne.

Testowanie rozwiązań

Żeby przetestować rozwiązanie zadania należy wykonać polecenie: `python3 zad1.py`

Zadanie offline 1.

Szablon rozwiązania: `zad1.py`

Cesarzowa Bajtocji zgubiła w napisie s swój ulubiony palindrom. Cesarzowa nikomu nie mówiła jaki jest jej ulubiony palindrom i wiadomo jedynie, że jest bardzo długi oraz składa się z nieparzystej liczby liter alfabetu łacińskiego. Postanowiono odnaleźć zaginiony palindrom cesarzowej. W tym celu należy zaimplementować funkcję:

```
def ceasar( s )
```

która na wejściu otrzymuje słowo s (składające się wyłącznie z małych liter alfabetu łacińskiego) i zwraca długość najdłuższego spójnego podsłowa, które jest palindromem i którego długość jest nieparzysta. Użyty algorytm powinien być możliwie jak najszybszy. Proszę uzasadnić jego poprawność i oszacować złożoność obliczeniową.

Przykład. Dla słowa:

```
akontnoknonabcddcba
```

wynikiem jest 7 (`kontnok`; proszę zwrócić uwagę, że `abcddcba` jest dłuższym palindromem, ale jest długości parzystej więc na pewno nie jest zagubionym palindromem cesarzowej).