WSTĘP DO PROGRAMOWANIA

Laboratorium 10
Funkcje i struktury danych: zadania zaawansowane

Zadania przygotowała: dr Anna Nenca

Testy będą na kolejnych zajęciach; informacja o nich została tu zawarta jedynie dla kompletności zadania. W tym tygodniu proszę pominąć tę część zadania – chyba że ktoś jest chętny, to w za tydzień będzie miał_a mniej pracy ;)

Zad. 1. Napisz program, który będzie zawierał małą bazę danych przechowujący dane planet: ich nazwy i gęstości. Dane powinny być przechowywane w liście zawierającej słowniki. Zdefiniuj kilka funkcji, które umożliwią wykonywanie operacji **CRUDS** oraz wykonaj ich testy:

a) funkcja, która umożliwi dodanie nowej planety do bazy (**CREATE**).

Nowa planeta powinna zostać dodana w taki sposób, aby wszystkie dane były posortowane rosnąco wg nazwy. Rozwiąż to zadanie bez sortowania danych.

Przykład: Mamy listę z danymi:

```
nazwa: 'Mars', gęstość: 5.427
nazwa: 'Ziemia', gęstość: 5.513
```

I chcemy dodać nową planetę o parametrach:

```
{nazwa: 'Wenus', gęstość: 5.204}
```

Zwrócona lista powinna być następująca (zgodnie z poniższą kolejnością):

```
nazwa: 'Mars', gęstość: 5.427
nazwa: 'Wenus', gęstość: 5.204
nazwa: 'Ziemia', gęstość: 5.513
```

Wejście: Zmienne:

- `name` – przechowuje napis będący nazwą planety,
- `density` – przechowuje liczbę nieujemną, która będzie gęstością planet w g/cm^3,
- `base` – przechowuje listę, która zawiera słowniki.

Wyjście: Lista, która zawiera informacje o planetach w bazie. Informacje o każdej planecie są przechowywane w osobnym słowniku. Dane są posortowane rosnąco wg nazw planet.

Warunki poprawności zadania: Upewnij się, że funkcja prawidłowo dodaje słownik i prawidłowo przechowuje informacje o nowej planecie. Upewnij się, że wszystkie dane są przechowywane zgodnie z porządkiem leksykograficznym wg nazw planet. Funkcja nie powinna sortować danych.

```
def addPlanet(name, denstiny , base):
def testAddPlanet():
```

b) funkcja, która zwróci krotkę zawierającą informację na temat nazwy i gęstości dla (pierwszej w liście) planety o podanej nazwie (zakładamy, że jest tylko jedna taka planeta) (**READ**).

Przykład: mamy listę z danymi:

```
nazwa: 'Mars', gęstość: 5.427
nazwa: 'Wenus', gęstość: 5.204
nazwa: 'Ziemia', gęstość: 5.513
```

Dla nazwy `'Ziemia'` funkcja powinna zwrócić:

```
('Ziemia', 5.513)
```

Wejście: Zmienne:

- `name` – przechowuje napis będący nazwą planety,
- `base` – przechowuje listę, która zawiera słowniki.

Wyjście: Krotka, która zawiera informację na temat nazwy i gęstości planety o podanej nazwie.

Warunki poprawności zadania: Upewnij się, że funkcja zwraca prawidłowe informacje.

```
def readPlanet(name, base):
def testReadPlanet():
```

c) funkcja, która umożliwi zmianę gęstości (pierwszej) planety o podanej nazwie (**UPDATE**).

Przykład: mamy listę z danymi:

```
nazwa: 'Mars', gęstość: 5.427
nazwa: 'Wenus', gęstość: 5.204
nazwa: 'Ziemia', gęstość: 5.513
```

Dla nazwy `'Wenus'` i gęstości `4.212`, funkcja powinna zwrócić listę z danymi:

```
nazwa: 'Mars', gęstość: 5.427
nazwa: 'Wenus', gęstość: 4.212
nazwa: 'Ziemia', gęstość: 5.513
```

Wejście: Zmienne:

- `name` – przechowuje napis będący nazwą planety,
- `density` – przechowuje liczbę nieujemną, która będzie gęstością planet w g/cm^3,
- `base` – przechowuje listę, która zawiera słowniki.

Wyjście: Lista, która zwiera dane z listy `base`, z gęstością planety o nazwie `name` zmienioną na `density`.

Warunki poprawności zadania: Upewnij się, że funkcja zwraca prawidłowe informacje.

```
def updatePlanet(name, denstiny , base):
def testUpdatePlanet():
```

d) funkcja, która umożliwi usunięcie planet, które mają mniejszą gęstość niż podana (**DELETE**).

Przykład: mamy listę z danymi:

```
nazwa: 'Mars', gęstość: 5.427
nazwa: 'Wenus', gęstość: 5.204
nazwa: 'Ziemia', gęstość: 5.513
```

Dla gęstości `5.500` funkcja powinna zwrócić listę z danymi:

```
nazwa: 'Ziemia', gęstość: 5.513
```

Wejście: Zmienne:

- `density` – przechowuje liczbę nieujemną, która będzie gęstością planet w g/cm^3,
- `base` – przechowuje listę, która zawiera słowniki.

Wyjście: Lista, która zwiera dane z listy `base` z usuniętymi elementami, które są mniejsze niż `density`.

Warunki poprawności zadania: Upewnij się, że funkcja zwraca prawidłowe informacje.

```
def deletePlanet(denstiny , base):
def testDeletePlanet():
```

e) Funkcja zwracająca indeks, na którym znajduje się (dowolna) planeta o podanej nazwie; jeżeli nie występuje taka planeta, funkcja zwraca wartość `-1` (**SEARCH**). W tym celu wyszukania planety zaimplementuj algorytm przeszukiwania binarnego.

Przykład: mamy listę z danymi:

```
nazwa: 'Mars', gęstość: 5.427
nazwa: 'Wenus', gęstość: 5.204
nazwa: 'Ziemia', gęstość: 5.513
```

Dla nazwy `'Wenus'`, powinno zostać zwrócone: `1` (ponieważ indeksujemy od 0); dla nazwy `'Pluton'` powinno zostać zwrócone: `-1`.

Wejście: Zmienne:

- `name` – przechowuje napis będący nazwą planety,
- `base` – przechowuje listę, która zawiera słowniki.

Wyjście: Indeks, na którym znajduje się planeta o nazwie `name` lub `-1` jeśli taka planeta nie jest zapisana w bazie.

Warunki poprawności zadania: Upewnij się, że funkcja zwraca prawidłową informację oraz że wyszukuje zgodnie z implementacją algorytmu przeszukiwania binarnego.

```
def searchPlanet(name, base):
def testSearchPlanet():
```