# Warunki

1. W implementacjach można korzystać tylko z elementarnych konstrukcji Python'a (funkcje, instrukcje warunkowe, pętle, `range`, klasy użyte do definiowania struktur danych). **Nie wolno korzystać ze słowników i zbiorów, itp.**

2. **Wolno korzystać z wbudowanego sortowania.**

3. Rozwiązania muszą być efektywne obliczeniowo (także w zadaniach, w których nie podajemy wprost ograniczenia na złożoność obliczeniową). Zadania o zbyt wysokiej złożoności będą otrzymywały obniżone oceny (lub 0).

4. Rozwiązania zadań proszę umieszczać w załączonych plikach (`zadX.py`).

# Zadanie 1 (bitoniczny komiwojażer)

W problemie bitonicznego komiwojażera mamy daną listę miast:

```
C = [["Wrocław", 0, 2], ["Warszawa",4,3],
     ["Gdańsk", 2,4], ["Kraków",3,1]]
```

gdzie każde miasto jest opisane przez listę składających się z jego nazwy i współrzędnych $x$, $y$ jego pozycji na mapie (dla uproszczenia pomijamy krzywiznę Ziemi). Miasta mają parami różne współrzędne $x$, ale mogą być podane w dowolnej kolejności. Bitoniczny komiwojażer startuje w mieście o najmniejszej współrzędnej $x$ i jego celem jest odwiedzić wszystkie miasta przy założeniu, że tylko raz może zmienić kierunek z "w prawo" na "w lewo" (dokładny opis problemu przedstawiony jest na wykładzie).

Proszę zaimplementować funkcję verb—bitonicTSP( C )—, która wypisuje na ekran długość trasy oraz po kolei miasta, które powinien odwiedzić komiwojażer. W przypadku miast z przykładu powyżej wynikiem może być:

```
Wrocław, Kraków, Warszawa, Gdańsk, Wrocław
```

(proszę zwrócić uwagę, że wynik nie jest jednoznaczny).