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. Wolno korzystać z:**
   (a) wbudowanego sortowania,
   (b) wbudowanej kolejki (`deque` z biblioteki `collections`),
   (c) wbudowanej kolejki priorytetowej (`PriorityQueue` z biblioteki `queue`).
2. 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).
3. Rozwiązania zadań proszę umieszczać w załączonych plikach (`zadX.py`).

Zadanie 1 (cykl o minimalnej wadze)

Dany jest graf nieskierowany reprezentowany przez macierz sąsiedztwa G, w której pole $G[i][j]$ ma wartość równą długości krawędzi z wierzchołka i do wierzchołka j, lub -1 jeśli krawędzi nie ma (wszystkie długości są nieujemne). Proszę zaimplementować funkcję:

```
min_cycle(G)
```

która sprawdza czy G ma cykl i jeśli tak, to zwraca cykl o najmniejszej sumie długości tworzących go krawędzi. Jeśli cyklu nie ma, funkcja powinna zwrócić pustą listę. Jeśli cykl istnieje, to funkcja powinna zwrócić listę wierzchołków, które tworzą ten cykl.

Przykład

Dla grafu reprezentowanego przez macierz:

```
G = [[-1, 2,-1,-1, 1],
     [ 2,-1, 4, 1,-1],
     [-1, 4,-1, 5,-1],
     [-1, 1, 5,-1, 3],
     [ 1,-1,-1, 3,-1]]
```

wynikiem może być, między innymi:

```
[0,1,3,4]
```

czyli cykl o łącznej długości $2 + 1 + 3 + 1 = 7$. Cykl `[1,2,3]` ma długość 10 i nie jest prawidłowym rozwiązaniem.