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 (drzewa BST)

Dane jest drzewo BST realizowane przez węzły zdefiniowane jak poniżej:

```
class BSTNode:
  def __init__(self, key):
    self.key = key
    self.left = None
    self.right = None
    self.parent = None
```

Proszę zaimplementować funkcje `insert( root, key )` oraz `remove( root, key)`, które przyjmują na wejściu korzeń drzewa oraz pewien klucz, odpowiednio, wstawiają go do drzewa lub usuwają z niego:

1. Funkcja `insert` zwraca `True` jeśli klucza nie było w drzewie i udało się go wstawić. W przeciwnym razie—jeśli klucz już jest w drzewie—nie wstawia go ponownie i zwraca `False`.
2. Funkcja `remove` zwraca `True` jeśli udało się usunąć klucz z drzewa lub `False` jeśli danego klucza w drzewie nie było.

W przypadku funkcji `remove` dopuszczalne jest **kopiowanie wartości klucza zamiast przepinanina wskaźników** gdy usuwany klucz jest w węźle zawierającym dwójkę dzieci.