Szablon rozwiązania	`egzP4b.py`
Złożoność akceptowalna (1.5pkt):	O($n + q\log n$)
Złożoność wzorcowa (+2.5pkt):	O(qh), gdzie ***n*** to całkowita liczba wierzchołków drzewa BST, ***h*** to jego wysokość, a ***q*** to rozmiar tablicy T.

Dane jest drzewo BST opisane przez następujące klasy:

```
class Node:
   def __init__( self, key, parent ):
     self.left = None     # lewe poddrzewo
     self.right = None    # prawe poddrzewo
     self.parent = parent # rodzic
     self.key = key       # wartość wierzchołka
     self.x = None        # pole do wykorzystania przez studentów
```

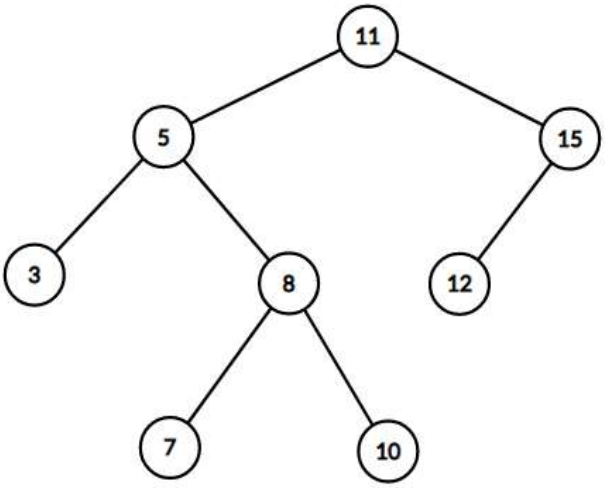
Mówimy, że wierzchołek takiego drzewa jest *ładny* jeżeli jego wartość jest średnią arytmetyczną wartości jego poprzednika oraz następnika. (Poprzednikiem wierzchołka nazywamy największy wierzchołek w drzewie BST mniejszy od niego, a następnikiem najmniejszy wierzchołek większy od niego. Oczywiście jako „najmniejszy”/”największy” rozumiemy wierzchołek o najmniejszej lub odpowiednio największej wartości). Zadanie polega na zaimplementowaniu funkcji:

```
def averagesum( T, root )
```

która dla danego drzewa o korzeniu *root* oraz tablicy wierzchołków T będących pewnymi wierzchołkami tego drzewa, zwraca sumę wartości wszystkich ładnych wierzchołków tego drzewa, jednocześnie będących elementami tablicy T. Można założyć, że wierzchołki o wartości najmniejszej oraz największej nie znajdują się w tablicy T, oraz, że nie zawiera ona powtórzeń. .

Rozważmy następujące dane:

```
w11 = Node(11, None)
w5 = Node(5, w11)
w11.left  = w5
w15 = Node(15, w11)
w11.right = w15
w3 = Node(3, w5)
w5.left = w3
w8 = Node(8, w5)
w5.right = w8
w12 = Node(12, w15)
w15.left  = w12
w7 = Node(7, w8)
w8.left   = w7
w10 = Node(10, w8)
w8.right  = w10
T = [ w5, w7, w8, w11, w12 ]
```



Wywołanie `averagesum( T, w11 )` powinno zwrócić wynik **16.** Wierzchołkami ładnymi są **5** (średnia arytmetyczna 3 i 7) oraz **11** (średnia arytmetyczna 10 i 12)