Warunki

1. Rozwiązania muszą być efektywne obliczeniowo (także w zadaniach, w których nie podajemy wprost ograniczenia na złożoność obliczeniową). Zadania o zbyt wysokiej złożoności będą oceniane na brak plusa.

2. **Wolno omawiać zadania (w tym pomysły na implementację), ale wyłącznie na forum w systmie UPEL. Nie wolno wymieniać kodu realizującego fragmenty algorytmu (ale wolno odpowiadać na pytania postaci "jak zrealizować tablicę dwuwymiarową" itp.).**

Zadanie 1 (Klocki)

Dany jest ciąg klocków $(K_1, \ldots, K_n)$. Klocek K_i zaczyna sie na pozycji a_i i ciągnie się do pozycji b_i (wszystkie pozycje to nieujemne liczby naturalne) oraz ma wysokość 1. Klocki układane są po kolei–jeśli klocek nachodzi na któryś z poprzednich, to jest przymocowywany na szczycie poprzedzajacego klocka). Na przykład dla klocków o pozycjach (1, 3), (2, 5), (0, 3), (8, 9), (4, 6) powstaje konstrukcja o wysokosci trzech klocków. Proszę zaimplementować funkcję `bricks(K)`, która oblicza wysokość powstałej konstrukcji (`K` to lista par (a_i, b_i), opisujących kolejne klocki).

Kod będzie uruchamiany. Na przykład wywołanie:

```
bricks( [ (1, 3), (2, 5), (0, 3), (8, 9), (4, 6)] )
```

powinno zwrócić liczbę 3.

Zadanie 2 (sumowanie przedziałów)

Proszę zaimplementować klasę `IntervalSums`, która przechowuje tablicę n liczb (na pozycjach od 0 do $n-1$), pozwala zmieniać zadane liczby oraz obliczać sumę liczb na pozycjach od i do j.

Państwa kod będzie uruchamiany. Proszę zaimplementować następujące funckcje w klase IntervalSums (można też dopisać inne klasy i/lub funkcje):

```
class IntervalSums:
    def __init__(self, n):
      # tworzy tablcę rozmiaru n, zainicjowaną zerami

    def set( self, i, val ):
      # zmienia zawartosc tablicy pod indeksem i na val

    def interval( self, i, j ):
      # zwraca sumę elementów tablice na pozycjach od i do j wlacznie
```

Przykładowe użycie klasy:

```
IS = IntervalSums(4)  # tworzy tablicę [0,0,0,0]
IS.set(0,10)          # [10,0,0,0]
IS.set(2,-2)          # [10,0,-2,0]
IS.set(3,1)           # [10,0,-2,1]
IS.interval(0,3)      # zwraca 10+0+(-2)+1 = 9
IS.interval(1,2)      # zwraca 0-2 = -2
```