Zadania o numerach 1 do 4 są przeznaczone do wykonania zdalnego, i przesłania ich rozwiązań w postaci kodów źródłowych oraz opisów (w jaki sposób podeszli Państwo do zadań oraz wnioski z ich wykonania). W ramach zadań 1, 2 i 4 dostępne języki programowania: Java, Python lub C#. W ramach zadania 3 można używać dowolnych języków i narzędzi.

Zadanie 1. **Problem semi-equilibrum**.
Tablica zawiera kolekcje liczb całkowitych (indeksowanie w tablicy od zera, w komórce o indeksie 0 mamy liczbę 1, indeks 2 wskazuje liczbę 5 itd)

1	2	5	2	1

Podać indeks elementu, który jest minimalnym przegięciem sumy lewostronnej nad prawostronną.
Przykład:
dla indeksu 0 – suma lewostronna: 1, suma prawostronna: 10
dla indeksu 1 – suma lewostronna: 3, suma prawostronna: 8
dla indeksu 2 – suma lewostronna: 8, suma prawostronna: 3
Wynikiem w tym przypadku jest indeks 2 (indeks przegięcia).
Napisać algorytm identyfikujący indeks przegięcia w złożoności O(n).

Zadanie 2. **Mechanizm weryfikacji poprawności domknięcia nawiasów.**
Przykład: (a(c) – ciąg niepoprawny
Przykład:)b – ciąg niepoprawny
Przykład: (b(s)) – ciąg poprawny
Funkcja przyjmuje ciąg znaków alfanumerycznych uzupełniony o nawiasy (lub), natomiast ma zwracać flagę: true/false. Cały algorytm ma mieć złożoność O(n). Dla podanych powyżej przykładów napisać testy jednostkowe.

Zadanie 3. **Wykrywanie spamu za pomocą algorytmów uczenia maszynowego.**
Klasyczny problem klasyfikacji maili jako spam lub nie-spam. W załączonych materiałach dostaliście katalog spambase (zawiera dane w formacie csv oraz opis atrybutów, specyfikację całej bazy). W pliku spambase.data każdy rekord to opis jednego maila za pomocą 57 atrybutów, a ostatnim 58 polem w rekordzie jest klasa (1-spam; 0-non-spam). Waszym zadaniem jest używając dowolnego języka (Java, C#, R, Python lub środowiska jak Weka, RapidMiner itp.) zbudować model klasyfikatora binarnego do klasyfikacji spam/nie-spam, oraz zbadać jego dokładność korzystając z metodyki ewaluacji 10-fold-cross-validation. Możecie użyć dowolnego algorytmu klasyfikacji, ale wyżej punktowane będą te metody, które prezentują lepszą dokładność wynikającą z przeprowadzonego 10-fold-cross-validation.

Zadanie 4. **Drzewa binarne.**
Dane jest drzewo binarne składające się z N wierzchołków. Mówimy, że wierzchołek T jest lewym potomkiem (lub odwrotnie, prawym), jeżeli wartość atrybutu l (lub odwrotnie, r) jest wartością różną od `null`. Ścieżką zstępującą nazywamy ciąg wierzchołków drzewa, którego każdy kolejny wierzchołek jest potomkiem poprzedniego. Długością ścieżki zstępującej jest liczba wierzchołków przez które przebiega minus 1.

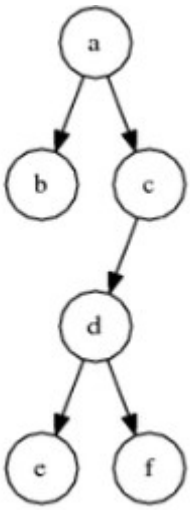
Mając podaną następującą deklarację

```
class Tree{
      public int x;
      public Tree l;
      public Tree r;
}
```

napisz funkcję w klasie `Solution`

```
class Solution {
      public int solution(Tree t);
}
```

która dla zadanego drzewa binarnego zwraca długość maksymalnej ścieżki zstępującej, która biegnie zawsze w lewo lub zawsze w prawo.
Np., dla następującego drzewa



poprawną odpowiedzią jest 2. Najdłuższa taka ścieżka startuje w wierzchołku *c* i zstępuje w lewo.

Załóż, że

- *N* jest liczbą całkowitą z przedziału [1..10000]
- wysokość drzewa *T* jest liczbą całkowitą z przedziału [0..800]
- drzewo *T* jest poprawnym drzewem binarnym

Wymagana złożoność

- oczekiwana pesymistyczna złożoność czasowa to *O(N)*
- oczekiwana pesymistyczna złożoność pamięciowa to *O(N)*