Imię i nazwisko: .... **WdI, Egzamin, 12.02.2020**
Indeks:..

Zadanie 1. [20] Niech X będzie tablicą liczb naturalnych dodatnich rozmiaru n, *niech* $k \leq n$ będzie dodatnią liczbą naturalną. Twoje zadanie polega na napisaniu efektywnego algorytmu, który znajdzie ciąg sąsiednich elementów w tablicy X, którego suma jest jak najbliższa sumie wszystkich elementów podzielonej przez k.

Formalnie, rozważmy problem opisany poniższą specyfikacją

Wejście:
n, k – liczby całkowite dodatnie takie, że $k \leq n$
X – tablica liczb naturalnych dodatnich rozmiaru n

Wyjście: para liczb i, j taka, że $0 \leq i \leq j < n$ oraz

$$\left|\sum_{p=i}^{j} X[p] - \frac{1}{k}\sum_{p=0}^{n-1} X[p]\right| \leq \left|\sum_{p=a}^{b} X[p] - \frac{1}{k}\sum_{p=0}^{n-1} X[p]\right|.$$

dla każdych naturalnych a, b takich, że $0 \leq a \leq b < n$.

Uwagi
– W sytuacji, gdy liczba par i, j spełniających powyższe warunki jest większa niż jeden, wystarczy podać dowolną z tych par.
– Możesz przyjąć, że operator dzielenia w języku C/Python daje poprawny wynik, bez zaokrągleń do liczb całkowitych (kwestie zgodności typów danych nie będą oceniane).

Twoje zadanie:
(a) Napisz funkcję (lub algorytm) rozwiązującą powyższy problem.
(b) Opisz ideę Twojego rozwiązania.
(c) Uzasadnij poprawność Twojego rozwiązania.
(d) Oszacuj złożoność czasową Twojego rozwiązania.

Przykład
Dla danych:

– $n = 4$, $k = 2$, $X = [\ 5, 7, 2, 4\ ]$,

mamy $\sum_{p=0}^{n-1} X[p] = 18,$ a wynikiem działania algorytmu będzie para liczb i=1, j=2, gdyż $\left|\sum_{k=1}^{2} X[k] - \frac{1}{k}\sum_{k=0}^{n-1} X[k]\right| = \left|7 + 2 - \frac{1}{2} \cdot (5 + 7 + 2 + 4)\right| = 0.$

Uwaga:
Maksymalna liczba punktów możliwa do uzyskania za to zadanie zależy od złożoności czasowej, w szczególności:
– Złożoność O(n): 20 punktów.
– Złożoność O(n^2): 9 punktów.
– Złożoność O(n^3): 6 punktów.

Imię i nazwisko: .... **WdI, Egzamin, 12.02.2020**
Indeks:..

Zadanie 2. [20] Załóżmy, że w *n*-elementowym ciągu X[0], …, X[n – 1] żaden element nie występuje więcej niż jeden raz. *k-tym elementem* ciągu X[0], …, X[n – 1] nazywamy taki element X[i] tego ciągu, że liczba elementów w ciągu mniejszych lub równych X[i] jest równa *k*.

Poniżej prezentujemy ogólny (i nieprecyzyjny) opis idei rekurencyjnego algorytmu znajdującego *k*-ty element w *n*-elementowym ciągu X[0], …, X[n – 1]:

I. Jeśli n = k = 1: zwróć jedyny element ciągu i zakończ.
II. *m* ← liczba elementów mniejszych od X[0]
III. Jeśli m < k – 1: uruchom algorytm rekurencyjnie na ciągu złożonym z elementów większych lub równych X[0] i odpowiednio zmodyfikowanych wartościach parametrów n i k. Zwróć wynik wywołania rekurencyjnego i zakończ.
IV. Jeśli m = k – 1: zwróć X[0] i zakończ.
V. Jeśli m > k – 1: uruchom algorytm rekurencyjnie na ciągu złożonym z elementów mniejszych lub równych X[0] i odpowiednio zmodyfikowanych wartościach parametrów n i k. Zwróć wynik wywołania rekurencyjnego i zakończ.

Twoje zadanie:

(a) Zapisz precyzyjnie powyższy algorytm jako funkcję rekurencyjną w C lub Python.

(b) Ustal asymptotyczną złożoność czasową najgorszego przypadku przedstawionego algorytmu. Odpowiedź uzasadnij i zilustruj przykładami.

Uwaga.

W analizie złożoności możesz dla uproszczenia przyjąć, że algorytm jest wywoływany dla k = n div 2.

Imię i nazwisko: .... **WdI, Egzamin, 12.02.2020**
Indeks:..

Zadanie 3. [20] Dana jest następująca struktura reprezentująca wierzchołek w drzewie binarnym:

```
typedef struct node *pnode;          class TreeItem:
typedef struct node{                 def __init__(self,value):
        int val;                         self.val = 1
        pnode left;                      self.left = None
        pnode right;} snode;             self.right = None
```

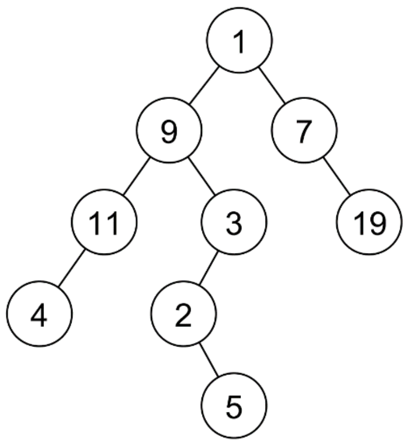
Twoje zadanie:
(a) Napisz funkcję `wypisz_ścieżkę(r, x)` realizującą następującą specyfikację:

Wejście: `r` – korzeń drzewa (typu `pnode` lub `TreeItem`),
`x` – liczba całkowita.
Wyjście:
1 – gdy element o wartości `x` występuje w drzewie o korzeniu `r`
0 – w przeciwnym przypadku
Dodatkowo, w sytuacji gdy wierzchołek o wartości `x` znajduje się w drzewie, należy wypisać na wyjściu całą ścieżkę z tego wierzchołka do korzenia.
W sytuacji, gdy istnieje kilka takich wierzchołków, można wybrać dowolny z nich.

(b) Podaj asymptotyczny czas działania swojego rozwiązania wraz z uzasadnieniem.

Uwaga: drzewo nie musi być drzewem BST.

Przykład
Dla poniższego drzewa:
- dla x=2 funkcja powinna zwrócić 1 i wypisać na wyjściu 2 3 9 1
- dla x=7 funkcja powinna zwrócić 1 i wypisać na wyjściu 7 1
- dla x=8 funkcja powinna zwrócić 0 i nic nie wypisywać



Imię i nazwisko: .... **WdI, Egzamin, 12.02.2020**
Indeks: ....

Zadanie 4. [20] Rozważmy gramatykę bezkontekstową G(N, T, P, S), gdzie
N = {S, X, Y, Z}
T = { a, b, c } a zbiór P składa się z produkcji:

$S \to X\ Y\ Z$
$X \to a\ X\ b,\ X \to \varepsilon$
$Y \to b,\ Y \to \varepsilon$
$Z \to b\ Z\ c,\ Z \to \varepsilon$

Uwaga: ε to słowo puste.

Twoje zadanie:

(a) [5] Dla każdego z poniższych (rodzin) napisów opisz sposób jego wyprowadzenia w gramatyce G lub uzasadnij, że nie należy on do języka L(G):

a a b b b c

a a b b c c c

$a^k\ b^{k+j}\ c^{k+j+p}$ gdzie *k, j, p* to liczby naturalne dodatnie

$a^k\ b^{k+j+p}\ c^j$ gdzie *k, j, p* to liczby naturalne dodatnie.

$a^{k+j}\ b^k\ c^{k+j+p}$ gdzie *k, j, p* to liczby naturalne dodatnie

(b) [15] Używając znanych Tobie notacji matematycznych, w tym notacji znanych ze wstępu do informatyki, precyzyjnie opisz język L(G), czyli zbiór słów, które można wyprowadzić w tej gramatyce. Następnie udowodnij, że zdefiniowana gramatyka G rzeczywiście definiuje dokładnie ten język, który został przez Ciebie zapisany.

Wskazówki:
– Opisując język L(G) prawdopodobnie będzie trzeba m.in. ustalić zależności między liczbą wystąpień liter a, b i c oraz kolejność, w jakiej one występują w słowie.
– Dowód wygodnie jest przeprowadzić np. metodą indukcji matematycznej; indukcja ze względu na długość słowa lub długość wyprowadzenia.

Imię i nazwisko: .... **WdI, Egzamin, 12.02.2020**
Indeks:..

Zadanie 5. [20] Na kwadratowej planszy składającej się z n^2 pól (n wierszy i n kolumn) możemy położyć co najwyżej n żetonów w taki sposób, że w każdym wierszu znajduje się co najwyżej 1 żeton i w każdej kolumnie znajduje się co najwyżej 1 żeton. Polu (i, j) na przecięciu i-tej kolumny i j-tego wiersza przypisana jest premia równa X[i][j]. Premia za rozłożone żetony jest równa sumie premii pól, na których żetony zostały położone.

Napisz funkcję (lub algorytm) realizującą następującą specyfikację:

Wejście: *n* – liczba naturalna,

X – tablica dwuwymiarowa liczb całkowitych taka, że X[i][j] jest równe premii dla pola (i, j)

s – liczba całkowita

Wyjście:

1 – jeśli istnieje poprawne rozmieszczenie żetonów, którego premia jest równa s
0 – jeśli NIE istnieje poprawne rozmieszczenie żetonów, którego premia jest równa s

Przykład

Dla *n*=5, *s*=10 i poniższej tablicy X istnieje poprawne rozmieszczenie żetonów zaznaczone szarym kolorem: (0, 1), (1, 2), (2, 0), (3, 3), (4, 4).

	0	1	2	3	4
0	1	1	2	1	1
1	2	1	1	1	1
2	0	2	1	1	1
3	0	1	1	2	1
4	2	0	1	1	2

Uwaga:
W tym zadaniu maksymalną liczbę punktów uzyskać można np. przedstawiając poprawne rozwiązanie stosujące przeszukiwanie z nawrotami.