Imię i nazwisko: ..... **WdI, Egzamin, 13.02.2023**
Indeks: .....

Zadanie 1. [20] Niech *X* będzie tablicą liczb całkowitych rozmiaru *n;* w tablicy *X* mogą znajdować się zarówno liczby dodatnie jak i ujemne.. Twoje zadanie polega na napisaniu algorytmu, który znajdzie największą sumę ciągu *sąsiednich* elementów w tablicy *X*. Formalnie, na wyjściu chcemy uzyskać liczbę *s* taką, że istnieją $0 \leq l \leq p < n$ dla których

$$s = \sum_{i=l}^{p} X[i]$$

oraz

$$s \geq \sum_{i=r}^{t} X[i]$$

dla każdych $0 \leq r \leq t < n$.

Wejście:
n – liczba całkowita dodatnia
X – tablica liczb całkowitych rozmiaru *n o indeksach* 0,..., *n* – 1

Wyjście:
s – największa suma ciągu sąsiednich elementów w tablicy *X*

Twoje zadanie:
(a) Napisz w pseudokodzie, C lub Pythonie 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 = 9, *X* = [–5, 7, 2, 4, –2, 1, 1, 1, –1],

wynikiem działania algorytmu będzie s = 7 + 2 + 4 + (–2) + 1 + 1 + 1 = 14 a dla danych

n = 9, *X* = [–5, –5, –5, –5, –1, –1, –3, –3, –2],

wynikiem działania algorytmu będzie *s* = –1.

Uwaga:
Maksymalna liczba punktów możliwa do uzyskania za to zadanie zależy od złożoności czasowej Twojego rozwiązania, w szczególności:
- Złożoność O(*n*): 20 punktów (a może nawet więcej ;)
- Złożoność O(n^2): 15 punktów,
- Złożoność O(n^3): 8 punktów.

Imię i nazwisko: ..... **WdI, Egzamin, 13.02.2023**
Indeks: .....

Imię i nazwisko: ..... **WdI, Egzamin, 13.02.2023**
Indeks: .....

Zadanie 2. [20] Rozważmy następującą funkcję:

```
funkcja Zagadka(A, x, l, p )
jeżeli l = p:
    jeżeli A[l] ≤ x: zwróć  A[l]
    w przeciwnym przypadku: zwróć 0
jeżeli l > p:  zwróć  0
s ← ( l+p ) / 2                    #dzielenie całkowite, zaokr. w dół
jeżeli ( l+p ) % 2 = 0: s ← s – 1  # % to operacja reszty z dzielenia
sL ← Zagadka( A, x, l, s )
sR ← Zagadka( A, x,s+1, p )
zwróć sL + sR
```

Twoje zadanie:

a) [4] Prześledź działanie funkcji *Zagadka* dla podanych poniżej wartości argumentów. Uzupełnij brakujące wartości w ostatniej kolumnie tabeli.

x	***l***	***p***	***A[l,…,p]***	***Zagadka(A, x, l, p)***
4	0	9	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]	
1	10	19	[2, 12,13, 14, 15, 1, 7, 8, 9, 10]	
21	0	4	[1, 12, 13, 4, 3]	
1	5	9	[1, 1, 1, 1, 1]	
5	6	6	[6]	

b) [6] Uzupełnij poniżej specyfikację funkcji Zagadka:

Wejście:
x – liczba całkowita,
l, p – liczby całkowite nieujemne, takie że $l \leq p$,
A[l,...,p] – część tablicy wypełnionej liczbami całkowitymi

Wyjście:

c) [4] Uzupełnij zależność rekurencyjną opisującą asymptotyczną złożoność czasową funkcji *Zagadka* dla $n = p - l + 1$:

$T(n) = 1$ dla $n < 2$

$T(n) =$ .....dla parzystego $n \geq 2$

$T(n) =$ .....dla nieparzystego $n \geq 2$

d) [6] Ustal i podaj zwartą postać funkcji T opisującej asymptotyczną złożoność czasową funkcji *Zagadka* rozwiązując zależność rekurencyjną z poprzedniego podpunktu, np. metodą podstawienia. Wystarczy, że uzyskasz rozwiązanie dla n postaci $n=2^k$ i naturalnego k.

Imię i nazwisko: ..... **WdI, Egzamin, 13.02.2023**
Indeks: .....

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

```
typedef struct node *pnode;
typedef struct node{
        int val;
        pnode left;
        pnode right;} snode;
```

```
class TreeItem:
def __init__(self,value):
    self.val = value
    self.left = None
    self.right = None
```

Twoje zadanie:

Przyjmujemy, że wierzchołek drzewa jest na *poziomie i* jeśli jego odległość od korzenia jest równa *i*. W szczególności korzeń drzewa jest na poziomie 0 a jego dzieci na poziomie 1. W wierzchołkach drzewa zapisane są też wartości – liczby całkowite w polu `val` każdego wierzchołka. *Sumą parzysto–nieparzystą* nazywać będziemy różnicę sumy wartości w wierzchołkach na parzystych poziomach i sumy wartości w wierzchołkach na poziomach nieparzystych.

(a) Napisz w pseudokodzie, C lub Pythonie algorytm/funkcję `sumaPN( r )` realizującą następującą specyfikację:

Wejście: `r` – korzeń drzewa (typu `pnode` lub `TreeItem`),

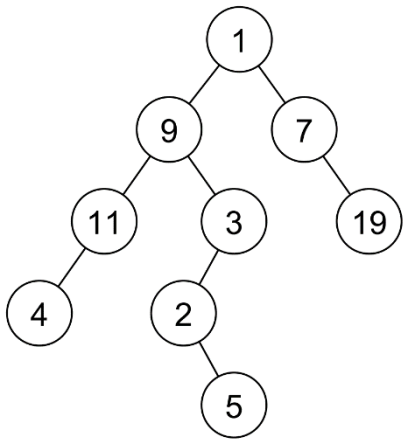
Wyjście: `s` – wartość sumy parzysto–nieparzystej drzewa o korzeniu `r`

(b) Podaj asymptotyczny czas działania swojego rozwiązania. Uzasadnij jego poprawność i podane przez Ciebie oszacowanie na czas działania.

Przykład
Dla poniższego drzewa suma parzysto–nieparzysta jest równa

$$(1 + 11 + 3 + 19 + 5) - (9 + 7 + 4 + 2) = 39 - 22 = 17$$

a dla drzewa pustego suma parzysto–nieparzysta jest równa 0.



Imię i nazwisko: ..... **WdI, Egzamin, 13.02.2023**
Indeks: .....

Imię i nazwisko: ..... **WdI, Egzamin, 13.02.2023**
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 → a a X, S → b, S → Y
X → b S
Y → c Z
Z → S c

Twoje zadanie:

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

a a b b

c c c a a b b c

a b c a a b b c

c a a b a a b b c

c a a b a b b c

(b) [5] Dla każdego z poniższych zdań

I. Liczba wystąpień litery a w **każdym** słowie z języka $L(G)$ spełnia warunki:.....

II. Liczba wystąpień litery b w **każdym** słowie z języka $L(G)$ spełnia warunki: .....

III. Liczba wystąpień litery c w **każdym** słowie z języka $L(G)$ spełnia warunki: .....

wybierz spośród (i) – (v) poprawne opcje:

(i) jest równa liczbie wystąpień litery b w tym słowie

(ii) jest równa 1 lub **nie** jest większa od liczby wystąpień litery a w tym słowie

(iii) jest podzielna przez 3

(iv) jest parzysta

(v) jest dokładnie o 2 mniejsza od dwukrotności liczby wystąpień jednej z dwóch pozostałych liter zbioru T w tym słowie.[1]

(c) [10] Rozważmy gramatykę G_1(N, T, P_1, S) otrzymaną w ten sposób, że P_1 to zbiór produkcji uzyskany przez usunięcie produkcji S → Y ze zbioru P. Używając znanych Tobie notacji matematycznych, w tym notacji znanych ze wstępu do informatyki, precyzyjnie opisz język L(G_1), czyli zbiór słów, które można wyprowadzić w gramatyce G_1. Następnie udowodnij, że zdefiniowana gramatyka G_1 rzeczywiście definiuje dokładnie ten język, który został przez Ciebie zapisany.

Wskazówka.
W pełni formalny dowód wygodnie jest przeprowadzić np. metodą indukcji matematycznej; indukcja ze względu na długość słowa lub długość wyprowadzenia.

[1] Na przykład dla zdania I. sformułowanie (v) należy interpreteować tak: Liczba wystąpień litery a jest dokładnie o 2 mniejsza od dwukrotności liczby wystąpień litery b lub liczba wystąpień litery a jest dokładnie o 2 mniejsza od dwukrotności liczby wystąpień litery c.

Imię i nazwisko: ..... **WdI, Egzamin, 13.02.2023**
Indeks: .....

Imię i nazwisko: ..... **WdI, Egzamin, 13.02.2023**
Indeks: .....

Zadanie 5. [20] W zbiorze *Z* złożonym z *n* agentów ponumerowanych od 0 do *n* – 1 niektórzy są ze sobą nawzajem *zaprzyjaźnieni*. Mając informację o tym, które pary agentów są zaprzyjaźnione należy ustalić czy możliwe jest wybranie podzbioru *P* złożonego z *k* agentów ze zbioru *Z* w taki sposób, że każdy agent ze zbioru *Z* jest w zbiorze *P* lub jest zaprzyjaźniony z co najmniej jednym agentem ze zbioru *P*. Informacje o zaprzyjaźnionych parach agentów zapisane są w dwuwymiarowej *tablicy zaprzyjaźnień* *A*[*n*][*n*] w taki sposób, że jeśli agenci *i* oraz *j* są zaprzyjaźnieni to *A*[*i*][*j*] = *A*[*j*][*i*] = 1 a w przeciwnym razie *A*[*i*][*j*] = *A*[*j*][*i*] = 0. Ponadto *A*[*i*][*i*] = 1 dla każdego $1 \leq i \leq n$, czyli każdy agent jest zaprzyjaźniony „sam ze sobą" ;)

Twoje zadanie

I. Napisz algorytm w pseudokodzie, C lub Pythonie realizujący poniższą specyfikację.

Specyfikacja

Wejście:

n, k – liczby naturalne dodatnie takie, że $\boldsymbol{n \geq k > 0}$
A[*n*][*n*] – tablica zaprzyjaźnień

Wyjście:

$\boldsymbol{z_1 \cdots z_k}$– ciąg parami różnych numerów agentów taki, że każdy agent należy do zbioru $\{\boldsymbol{z_1 \cdots z_k}\}$ lub jest zaprzyjaźniony z jakimś agentem z tego zbioru, jeśli zbiór spełniający takie warunki istnieje
$\boldsymbol{-1}$ – w przeciwnym przypadku.

II. Opisz ideę działania swojego algorytmu. Uzasadnij jego poprawność.

Uwagi.

1. Maksymalną liczbę punktów za to zadanie można uzyskać np. za poprawne i efektywne rozwiązanie oparte na przeszukiwaniu z nawrotami.
2. Za rozwiązanie łatwiejszej wersji zadania z wyjściem:

   Wyjście:

   $\boldsymbol{1}$– gdy istnieje pozbiór *P* zbioru *Z* złożony z *k* agentów taki, że każdy agent z *Z* należy do zbioru *P* lub jest zaprzyjaźniony z jakimś agentem z tego zbioru,
   $\boldsymbol{-1}$ – w przeciwnym przypadku.

   uzyskać można **15 punktów**!

Przykład.

Rozważmy *n* = 6 oraz następującą *tablicę zaprzyjaźnieni A*:

	0	1	2	3	4	5
	1	**1**	**0**	**0**	**0**	**0**
1	**1**	**1**	**1**	**0**	**1**	**1**
2	**0**	**1**	**1**	**0**	**0**	**0**
3	**0**	**0**	**0**	**1**	**1**	**1**
4	**0**	**1**	**0**	**1**	**1**	**1**
5	**0**	**1**	**0**	**1**	**1**	**1**

Wówczas:

- dla $k = 2$ poprawnym ciągami na wyjściu jest np. 1 5 (lub 1 4 lub 1 3), ponieważ każdy agent jest zaprzyjaźniony z agentem 1 lub agentem 5 (podobnie, każdy agent jest zaprzyjaźniony z agentem 1 lub agentem 4; każdy agent jest zaprzyjaźniony z agentem 1 lub agentem 3);
- dla $k = 1$ wartość na wyjściu jest równa $\mathbf{-1}$, ponieważ żaden agent nie jest zaprzyjaźniony ze wszystkimi pozostałymi agentami.