



**„POLTELEINFO”**  
**Ogólnopolska Olimpiada Liderów Telekomunikacji i Informatyki**  
**Rok szkolny 2024/2025**

**Zadania dla grupy informatycznej na zawody II stopnia**

**Instrukcja dla uczestnika**

1. Czas trwania zawodów: 120 minut.
2. II stopień Olimpiady zawiera 5 zadań.
3. Należy podać poprawną odpowiedź wraz z tokiem rozwiązania.
4. Za każdą prawidłową odpowiedź uzyskuje się maksymalnie 10 punktów. Maksymalna liczba punktów do zdobycia za 5 zadań to 50 punktów.
5. Można korzystać z przyborów do pisania, kalkulatorów i tablic matematycznych oraz rozdawanych kart czystopisu i brudnopisu. Korzystanie z notebooków, tabletów, telefonów komórkowych, smartfonów, smartwatchy, kalkulatorów programowalnych, itp. jest zabronione.

**Życzymy powodzenia!**

**Zadanie 1.**

Zadanie składa się z 10 pytań zamkniętych wielokrotnego wyboru. W każdym należy wybrać 1-3 prawidłowych odpowiedzi. Odpowiedzi należy nanieść na tabelkę znajdującą się pod niniejszym zadaniem. Odpowiedzi zaznaczaj w tabeli X.

Uwaga tylko odpowiedzi znajdujące się w tabeli będą oceniane!

**1.1. Test Touringa:**

- a) został zaproponowany w II połowie XX wieku
- b) miał pozwalać na odróżnienie sztucznej inteligencji od naturalnej (ludzkiej)
- c) został łatwo zdany już kilka lat po propozycji

**1.2. Jakiego rodzaju sieci neuronowe są najczęściej stosowane w rozpoznawaniu obrazów?**

- a) splotowe
- b) sztuczne
- c) rekurencyjne

**1.3. Proces uczenia korzysta z danych:**

- a) treningowych

- b) walidacyjnych
- c) testowych

**1.4.** Macierz pomyłek jest bazą do obliczenia metryki:

- a) F1-score
- b) Czułość (Recall)
- c) Dokładność (Accuracy)

**1.5.** Proces ekstrakcji cech:

- a) może być realizowany za pomocą sztucznej sieci neuronowej
- b) pozwala na redukcję złożoności problemu
- c) powoduje utratę danych istotnych z punktu widzenia problemu

**1.6.** Podstawowym elementem sztucznej sieci neuronowej nie są:

- a) Gradienty
- b) Warstwy
- c) Neurony

**1.7.** Które z poniższych algorytmów nie wpisują się w dziedzinę uczenia maszynowego?

- a) Algorytm DFS (Depth-First Search)
- b) Drzewa decyzyjne (Decision Trees)
- c) K-najbliższych sąsiadów (K-Nearest Neighbors)

**1.8.** MSE (Mean Squared Error) jest używany do oceny systemów:

- a) regresji
- b) klasyfikacji
- c) segmentacji

**1.9.** Które z poniższych technik pomagają uniknąć przeuczenia (overfittingu) w modelach ML?

- a) Regularyzacja
- b) Zwiększenie ilości danych treningowych
- c) Zwiększenie liczby epok treningowych

**1.10.** Niedopasowanie do danych uczących (underfitting) modelu może być wynikiem:

- a) Zbyt prostej architektury sieci
- b) Zbyt niskiej jakości danych treningowych
- c) Zbyt dużej liczby danych treningowych

Tabela odpowiedzi do zadania 1.

Pytanie	Odpowiedzi		
1.1	<input checked="" type="radio"/> A	<input checked="" type="radio"/> B	C
1.2	<input checked="" type="radio"/> A	<input checked="" type="radio"/> B	C
1.3	<input checked="" type="radio"/> A	<input checked="" type="radio"/> B	C
1.4	<input checked="" type="radio"/> A	<input checked="" type="radio"/> B	<input checked="" type="radio"/> C
1.5	<input checked="" type="radio"/> A	<input checked="" type="radio"/> B	C
1.6	<input checked="" type="radio"/> A	B	C
1.7	<input checked="" type="radio"/> A	B	C
1.8	<input checked="" type="radio"/> A	B	C
1.9	<input checked="" type="radio"/> A	<input checked="" type="radio"/> B	C
1.10	<input checked="" type="radio"/> A	<input checked="" type="radio"/> B	C

## Zadanie 2.

Poniżej przedstawiono kod w języku Python.

**Zadanie 2.1.** Podaj, co będzie efektem działania tego kodu.

**Zadanie 2.2.** Uzasadnij każdy wynik, odnosząc się do zasad programowania obiektowego.

```

1 class A:
2     x = 1
3     def __init__(self):
4         self.y = 2
5
6     def __str__(self):
7         return f'({ self.x }, { self.y })'
8
9 class B(A):
10     y = 3
11     def __init__(self):
12         self.x = 2
13
14 class C(A):
15     y = 5
16     def __init__(self):
17         x = 6
18
19 class D(B, C):
20     def __init__(self):
21         pass
22
23 print(A())
24 print(B())
25 print(C())
26 print(D())

```

## **Zadanie 2.1**

Wynik dla print A - (1,2)

Wynik dla print B - (2,3)

Wynik dla print C - (1,5)

Wynik dla print D - (1,3)

## **Zadanie 2.2**

Klasy w tym kodzie ilustrują podstawowe zasady programowania obiektowego, takie jak enkapsulacja, polimorfizm, dziedziczenie oraz działanie konstruktorów.

Klasa A:

Enkapsulacja – posiada zarówno zmienną klasową x, jak i zmienną instancji y.

Konstruktor (`__init__`) – inicjalizuje obiekt i przypisuje wartości początkowe do atrybutów instancji.

Polimorfizm – metoda specjalna `__str__` pozwala na dostosowanie sposobu wyświetlania obiektu, co jest przydatne np. podczas debugowania.

Klasa B:

Dziedziczenie – klasa B dziedziczy po A, co oznacza, że przejmuje jej metody i atrybuty.

Konstruktor (`__init__`) – nadpisuje wartość x, ale nie wywołuje `super().__init__()`, przez co y nie jest inicjalizowane.

Klasa C:

Dziedziczenie – klasa C dziedziczy po A, co zapewnia dostęp do jej metod i atrybutów.

Błąd logiczny – `x = 6` w konstruktorze to lokalna zmienna, a nie atrybut instancji (`self.x`), co oznacza, że nie modyfikuje wartości klasy A ani nie tworzy nowego atrybutu instancji.

Klasa D:

Dziedziczenie wielokrotne i MRO – Python używa Method Resolution Order (MRO) do ustalenia kolejności przeszukiwania klas nadrzędnych:  $D \rightarrow B \rightarrow C \rightarrow A$ .

Konstruktor – nie wywołuje `super().__init__()`, więc obiekt D nie będzie miał `self.x` ani `self.y`, ponieważ żadna z klas nadrzędnych ich nie inicjalizuje.

### Zadanie 3.

Podciąg to dowolny zbiór wybranych elementów ciągu, które zachowują oryginalną kolejność. Wyróżniamy podciągi spójne (ciągłe), które składają się z kolejnych, sąsiadujących elementów oryginalnej tablicy oraz podciągi niespójne (dowolne), które mogą zawierać dowolne elementy, ale muszą występować w tej samej kolejności co w oryginalnym ciągu.

Dany jest program napisany w języku C++ zliczający podciągi N elementowego ciągu arr, których suma elementów jest podzielna przez liczbę K. Przeanalizuj jego działanie i odpowiedz na poniższe pytania.

#### Listing 1

```
#include <iostream>
#include <fstream>
#include <vector>

using namespace std;

int countSubsequences(int N, int K, vector<int>& arr) {
    int i = 0;
    vector<int> dp(K, 0);
    dp[0] = 1;

    while(i < N) {
        int num = arr[i];
        vector<int> new_dp = dp;

        for (int j = 0; j < K; j++) {
            int new_mod = (j + num % K + K) % K;
            new_dp[new_mod] += dp[j];
        }

        dp = new_dp;
        i++;
    }

    return dp[0]-1;
}

int countCoherent(int N, int K, vector<int>& arr) {
    vector<int> dp(K, 0);
    dp[0] = 1;

    int count = 0;
    int prefixSum = 0;

    for (int num /*A*/) {
        prefixSum = (prefixSum + num % K + K) % K;
```

```

        count += dp[prefixSum];
        dp[prefixSum]++;
    }
    /*B*/
}

int main() {
    ifstream inputFile;
    try{
        inputFile.open("data.txt");
        if(!inputFile)
            throw runtime_error("File not found!");

        int N, K;
        vector<int> arr;
        int number;
        bool isConfigLine = true;

        while (inputFile >> N >> K){
            arr.clear();
            for (int i = 0 ; i < N ; i++){
                if(!(inputFile >> number))
                    throw runtime_error("Invalid data
format");
                arr.push_back(number);
            }

            cout << countSubsequences(N, K, arr) << endl;
        }

        inputFile.close();
    }catch(exception e){
        cout << "Error: " << e.what() << endl;
        return 1;
    }
    return 0;
}

```

### Zadanie 3.1

Jaką złożoność obliczeniową (w notacji  $O$ ) ma zaprezentowany w funkcji *countSubsequences()* algorytm? Odpowiedź uzasadnij.

#### Odpowiedź:

$O(N*K)$  [lub  $O(NK)$ ,  $O(KN)$ ,  $O(K*N)$ ].

### Zadanie 3.2

Dlaczego w linii `int new_mod = (j + num % K + K) % K;` dodajemy wartość  $+K$  przed obliczeniem modulo? Odpowiedź uzasadnij odnosząc się do zasad matematyki i programowania.

**Odpowiedź:**

W matematyce reszta z dzielenia liczby ujemnej przez liczbę dodatnią może przyjmować liczbę ujemną lub dodatnią np.

$-5\%3 = -1$  reszty  $-2$  [bo  $-1*3-2 = -3-2 = -5$ ]

lub  $-5\%3 = -2$  reszty  $1$  [bo  $-2*3+1 = -6+1 = -5$ ]

a dodanie wartości  $K$  powoduje zwrócenie zawsze wartości dodatniej.

`new_mod` jest wykorzystywany jako indeks tablicy, który nie może być ujemny.

**Zadanie 3.3**

Jakie liczby zwróci funkcja `countSubsequences()` jeśli w pliku znajdują się dane:

data.txt	
4	3
3	6 2 9
5	3
7	5 -8 11 0

**Odpowiedź:**

Dla zestawu 1: 7.

Dla zestawu 2: 11.

**Zadanie 3.4**

Dla danych  $N=5$ ,  $K=4$ , `arr = [4, 2, 2, 8, 6]` wypisz znalezione podciągi spójne, których suma elementów jest podzielna przez  $K$ .

**Odpowiedź:**

[4], [8], [2, 2], [4, 2, 2], [2, 2, 8], [2, 8, 6], [4, 2, 2, 8]

lub [ $\emptyset$ ], [4], [8], [2, 2], [4, 2, 2], [2, 2, 8], [2, 8, 6], [4, 2, 2, 8].

**Zadanie 3.5**

Uzupełnij kod funkcji `countCoherent(int N, int K, vector<int>& arr)` w punktach `/*A*/` i `/*B*/` tak aby funkcja zliczała podciągi spójne, których suma elementów jest podzielna przez  $K$ .

**Odpowiedź:**

`/*A*/` „arr”

/\*B\*/ „return count;”

#### Zadanie 4. Bramki logiczne

**Zadanie 4.1** Oceń prawdziwość podanych zdań

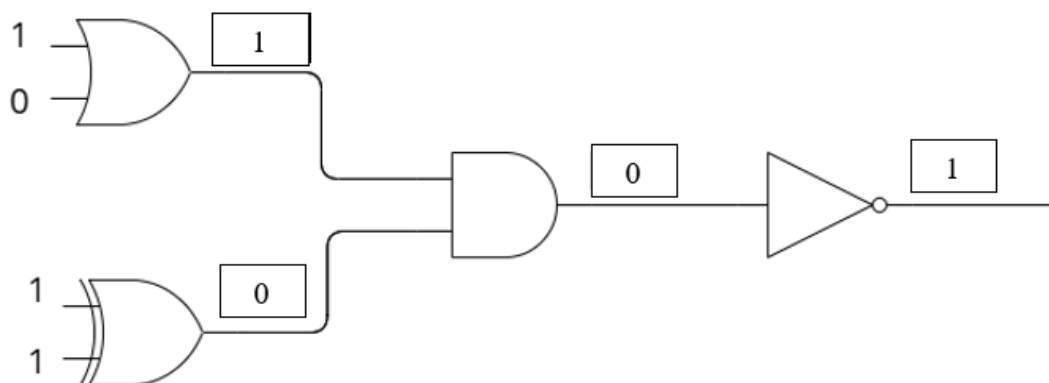
##### Rozwiązanie

a). Oceń prawdziwość podanych zdań

	Prawda	Fałsz
Bramka logiczna może być dwustanowa lub trójstanowa.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Bramka XOR zawsze zwraca stan wysoki, jeśli na jej wejściach pojawia się co najmniej jeden stan wysoki.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Bramka XNOR daje wynik wysoki tylko wtedy, gdy liczba jedynek na wejściu jest parzysta.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Bramka NOR jest połączeniem bramek OR i NOT.	<input checked="" type="checkbox"/>	<input type="checkbox"/>

**Zadanie 4.2.** Przeanalizuj układ i podaj wartości sygnałów na wyjściu z poszczególnych bramek. W tym celu odpowiednie wartości wpisz w podanych prostokątnych polach.

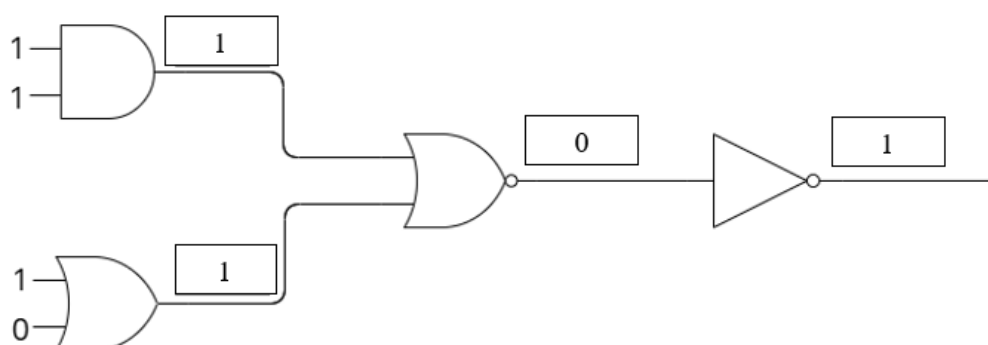
##### Odpowiedź:



**Zadanie 4.3.** Przeanalizuj układ i podaj wartości sygnałów na wyjściu z poszczególnych bramek. W tym celu odpowiednie wartości wpisz w podanych prostokątnych polach.

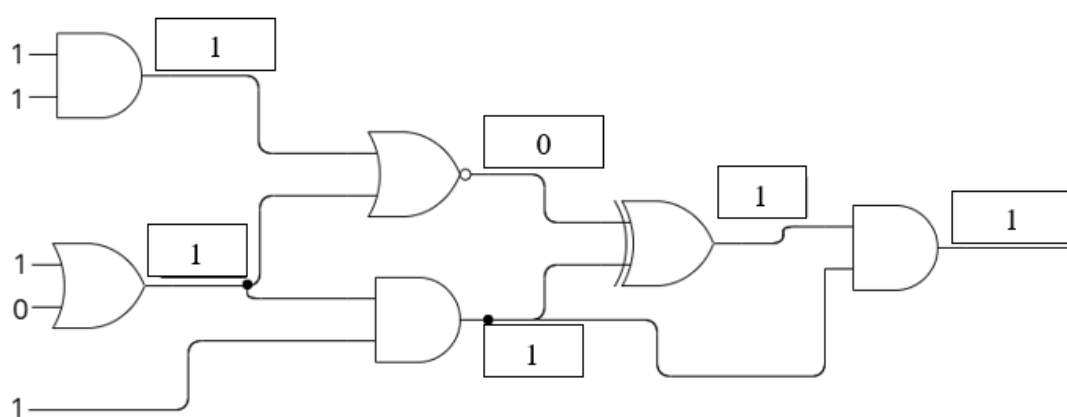


**Odpowiedź:**



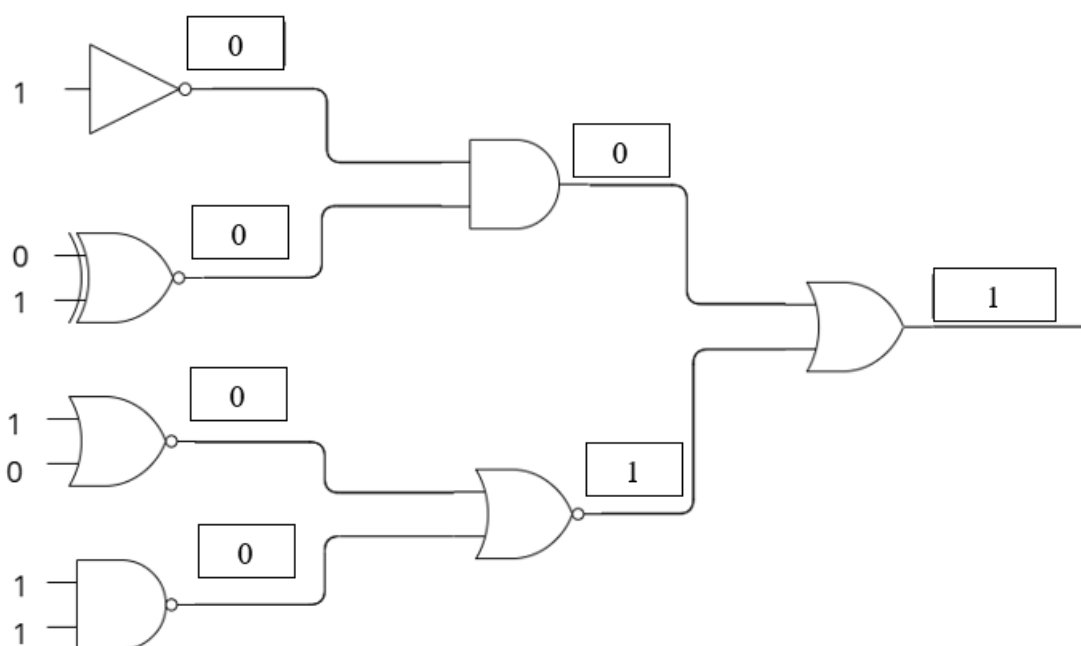
**Zadanie 4.4.** Przeanalizuj układ i podaj wartości sygnałów na wyjściu z poszczególnych bramek. W tym celu odpowiednie wartości wpisz w podanych prostokątnych polach.

**Odpowiedź:**



**Zadanie 4.5.** Przeanalizuj układ i podaj wartości sygnałów na wyjściu z poszczególnych bramek. W tym celu odpowiednie wartości wpisz w podanych prostokątnych polach.

**Odpowiedź:**



**Zadanie 5.**

Miasto planuje wdrożenie sieci sensorów IoT do monitorowania jakości powietrza. Każdy sensor IoT może przesyłać dane do serwera bezpośrednio lub za pośrednictwem innych sensorów. Każda transmisja ma swój koszt energetyczny.

**Założenia**

- Sensory są połączone w sieć i mogą przekazywać dane między sobą.
- Każde połączenie między dwoma sensorami ma określony koszt energetyczny.
- Chcemy znaleźć **najtańszą ścieżkę** przesyłu danych z **każdego sensora do serwera centralnego**.

**Wejście**

1. Liczba sensorów **n** i liczba połączeń **m**.
2. Lista **m** połączeń w formacie:

sensorA sensorB koszt

Gdzie **sensorA** i **sensorB** to połączone sensory, a **koszt** to liczba określająca koszt energetyczny transmisji.

3. Numer sensora pełniącego rolę **serwera centralnego**.

**Wejście**

$n = 5$ ,  $m = 6$

1 2 4

1 3 2

2 4 7

3 4 3

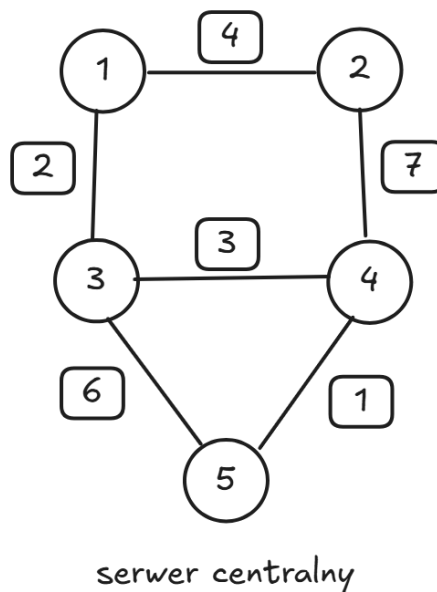
3 5 6

4 5 1

Serwer centralny: 5

**Zadanie 5.1** Narysuj graf połączeń pomiędzy sensorami.

**Odpowiedź:**



**Zadanie 5.2** Określ najtańszą ścieżkę dojścia od każdego z sensorów do serwera centralnego.

W odpowiedzi uwzględnij trasy oraz ich koszt.

**Odpowiedź:**

<b>1 → 5</b>	Najtańsza 1—3—4—5 koszt 6
<b>2 → 5</b>	Najtańsza 2—4—5 koszt 8
<b>3 → 5</b>	Najtańsza 3—4—5 koszt 4
<b>4 → 5</b>	Najtańsza 4—5 koszt 1

**Zadanie 5.3** Wymień 2 algorytmy wyznaczania najkrótszej ścieżki.

**Odpowiedź:**

Algorytm Dijkstry, Algorytm Bellmana-Forda, A\*

**Zadanie 5.4** Dla jednego z podanych w zadaniu 5.3 algorytmu wyjaśnij zasadę jego działania uwzględniając: warunki początkowe oraz zasadę działania.

**Przykładowa odpowiedź:**

## Algorytm Dijkstry

Założenia początkowe:

- graf skierowany lub nieskierowany o nieujemnych wagach krawędzi

Zasada działania:

Algorytm rozpoczyna działanie od ustalonego wierzchołka początkowego.

Inicjalizacja tablicy odległości:

- Wierzchołkowi startowemu przypisujemy odległość 0, ponieważ już w nim jesteśmy.
- Wszystkie pozostałe wierzchołki otrzymują wartość  $\infty$ , gdyż jeszcze do nich nie dotarliśmy.

Na początku wszystkie wierzchołki należą do zbioru Q, oznaczającego wierzchołki nieprzetworzone. Następnie algorytm przebiega według następujących kroków:

1. Dopóki zbiór Q nie jest pusty:
  - Wybierz z Q wierzchołek o najmniejszym koszcie dotarcia, oznacz go jako v i usuń go ze zbioru.
  - Dla każdej krawędzi wychodzącej z v (oznaczmy ją jako k) wykonaj następujące operacje:
    - Oznacz wierzchołek na drugim końcu krawędzi k jako u.
    - Jeśli koszt dotarcia do u przez v i krawędź k jest mniejszy niż obecnie zapisany koszt dotarcia do u, to:
      - Zaktualizuj koszt dotarcia do u, sumując koszt dotarcia do v i wagę krawędzi k.
      - Ustaw v jako poprzednika wierzchołka u.