| | |
|---|---|
| *Rodzaj dokumentu:* | **Sprawozdanie za rok 2026** |
| *Egzamin:* | **Egzamin maturalny** |
| *Przedmiot:* | **Informatyka** |
| *Poziom:* | **Poziom rozszerzony** |
| *Termin egzaminu:* | 14 maja 2026 r. |
| *Data publikacji dokumentu:* | 25 września 2026 r. |

**WOJEWÓDZTWO LUBELSKIE**

Sprawozdanie zostało opracowane przez Centralną Komisję Egzaminacyjną we współpracy z okręgowymi komisjami egzaminacyjnymi.

**Centralna Komisja Egzaminacyjna**
ul. Józefa Lewartowskiego 6, 00-190 Warszawa
tel. 22 536 65 00
www.cke.gov.pl sekretariat@cke.gov.pl

**Okręgowa Komisja Egzaminacyjna w Gdańsku** (województwa: kujawsko-pomorskie, pomorskie)
ul. Na Stoku 49, 80-874 Gdańsk
tel. 58 320 55 90
www.oke.gda.pl komisja@oke.gda.pl

**Okręgowa Komisja Egzaminacyjna w Jaworznie** (województwo śląskie)
ul. Adama Mickiewicza 4, 43-600 Jaworzno
tel. 32 784 16 00
www.oke.jaworzno.pl sekretariat@oke.jaworzno.pl

**Okręgowa Komisja Egzaminacyjna w Krakowie** (województwa: lubelskie, małopolskie, podkarpackie)
os. Szkolne 37, 31-978 Kraków
tel. 12 683 21 01
www.oke.krakow.pl oke@oke.krakow.pl

**Okręgowa Komisja Egzaminacyjna w Łomży** (województwa: podlaskie, warmińsko-mazurskie)
Al. Legionów 9, 18-400 Łomża
tel. 86 473 71 20
www.oke.lomza.pl sekretariat@oke.lomza.pl

**Okręgowa Komisja Egzaminacyjna w Łodzi** (województwa: łódzkie, świętokrzyskie)
ul. Ksawerego Praussa 4, 94-203 Łódź
tel. 42 664 80 50
lodz.oke.gov.pl sekretariat@lodz.oke.gov.pl

**Okręgowa Komisja Egzaminacyjna w Poznaniu** (województwa: lubuskie, wielkopolskie, zachodniopomorskie)
ul. Gronowa 22, 61-655 Poznań
tel. 61 854 01 60
www.oke.poznan.pl sekretariat@oke.poznan.pl

**Okręgowa Komisja Egzaminacyjna w Warszawie** (województwo mazowieckie)
ul. Józefa Bema 87, 01-233 Warszawa
tel. 22 457 03 35
www.oke.waw.pl info@oke.waw.pl

**Okręgowa Komisja Egzaminacyjna we Wrocławiu** (województwa: dolnośląskie, opolskie)
ul. Tadeusza Zielińskiego 57, 53-533 Wrocław
tel. 71 785 18 94
www.oke.wroc.pl sekretariat@oke.wroc.pl

Serwis Komisji Egzaminacyjnych: egzaminy.gov.pl

## Spis treści

## Opis arkusza egzaminu maturalnego

W roku szkolnym 2025/2026 egzamin maturalny z informatyki został przeprowadzany na podstawie wymagań podstawy programowej określonych w rozporządzeniu Ministra Edukacji z dnia 28 czerwca 2024 r.[1]

Arkusz egzaminacyjny z informatyki na poziomie rozszerzonym zawierał ogółem 23 zadania w tym 21 zadań ujętych w 6 wiązkach tematycznych. Za rozwiązanie wszystkich zadań można było otrzymać 50 punktów.

Zadania sprawdzały wiadomości oraz umiejętności ujęte w trzech obszarach wymagań ogólnych:

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi

III. Posługiwanie się komputerem, urządzeniami cyfrowymi i sieciami komputerowymi, w tym: znajomość zasad działania urządzeń cyfrowych i sieci komputerowych oraz wykonywania obliczeń i programów.

Egzamin trwał 210 minut. Zdający przez cały czas trwania egzaminu korzystali z autonomicznego stanowiska komputerowego.

[1] Rozporządzenie Ministra Edukacji z dnia 28 czerwca 2024 r. zmieniające rozporządzenie w sprawie podstawy programowej kształcenia ogólnego dla liceum ogólnokształcącego, technikum oraz branżowej szkoły II stopnia (Dz.U. poz. 1019).

## Dane dotyczące populacji zdających

TABELA 1. ZDAJĄCY ROZWIĄZUJĄCY ZADANIA W ARKUSZU STANDARDOWYM*

| **Liczba zdających (Formuła 2023)** | | **419** |
|---|---|---|
| Zdający rozwiązujący zadania w arkuszu standardowym | z liceów ogólnokształcących | 196 |
| | z techników | 223 |
| | z branżowych szkół II stopnia | 0 |
| | ze szkół na wsi | 0 |
| | ze szkół w miastach do 20 tys. mieszkańców | 94 |
| | ze szkół w miastach od 20 tys. do 100 tys. mieszkańców | 185 |
| | ze szkół w miastach powyżej 100 tys. mieszkańców | 140 |
| | ze szkół publicznych | 389 |
| | ze szkół niepublicznych | 30 |
| | kobiety | 39 |
| | mężczyźni | 380 |
| | bez dysleksji rozwojowej | 338 |
| | z dysleksją rozwojową | 81 |

* Dane w tabeli dotyczą tegorocznych absolwentów.

Z egzaminu zwolniono 1 osobę − laureata Olimpiady Informatycznej.

TABELA 2. ZDAJĄCY ROZWIĄZUJĄCY ZADANIA W ARKUSZACH DOSTOSOWANYCH

| Zdający rozwiązujący zadania w arkuszach dostosowanych | z autyzmem, w tym z zespołem Aspergera | 10 |
|---|---|---|
| | słabowidzący | 0 |
| | niewidomi | 0 |
| | słabosłyszący | 0 |
| | niesłyszący | 0 |
| | z niepełnosprawnością ruchową spowodowaną mózgowym porażeniem dziecięcym | 0 |
| | z zaburzeniem widzenia barw | 0 |
| | **Ogółem** | **10** |

## Przebieg egzaminu

TABELA 3. INFORMACJE DOTYCZĄCE PRZEBIEGU EGZAMINU

| | | | |
|---|---|---|---|
| Termin egzaminu | | | 14 maja 2026 |
| Czas trwania egzaminu dla arkusza standardowego | | | 210 minut |
| Liczba szkół | | | 91 |
| Liczba zespołów egzaminatorów | | | - |
| Liczba egzaminatorów-weryfikatorów | | | - |
| Liczba egzaminatorów | | | - |
| Liczba obserwatorów[2] (§ 8 ust. 1) | | | |
| Liczba unieważnień[4] | w przypadku: | | |
| | art. 44zzv pkt 1 | stwierdzenia niesamodzielnego rozwiązywania zadań przez zdającego | - |
| | art. 44zzv pkt 2 | wniesienia lub korzystania przez zdającego w sali egzaminacyjnej z urządzenia telekomunikacyjnego | - |
| | art. 44zzv pkt 3 | zakłócenia przez zdającego prawidłowego przebiegu egzaminu | - |
| | art. 44zzw ust. 1 | stwierdzenia podczas sprawdzania pracy niesamodzielnego rozwiązywania zadań przez zdającego | - |
| | art. 44zzy ust. 7 | stwierdzenie naruszenia przepisów dotyczących przeprowadzenia egzaminu maturalnego | 1 |
| | art. 44zzy ust. 10 | niemożność ustalenia wyniku (np. zaginięcie karty odpowiedzi) | - |
| Liczba wglądów[3] (art. 44zzz) | | | 8 |

[2] Rozporządzenie Ministra Edukacji i Nauki z dnia 1 sierpnia 2022 r. w sprawie egzaminu maturalnego (Dz.U. z 2024 r. poz. 302, z późn. zm.).

[3] Ustawa z dnia 7 września 1991 r. o systemie oświaty (Dz.U. z 2025 r. poz. 881, z późn. zm.).

# Podstawowe dane statystyczne

## Wyniki zdających

WYKRES 1. ROZKŁAD WYNIKÓW ZDAJĄCYCH

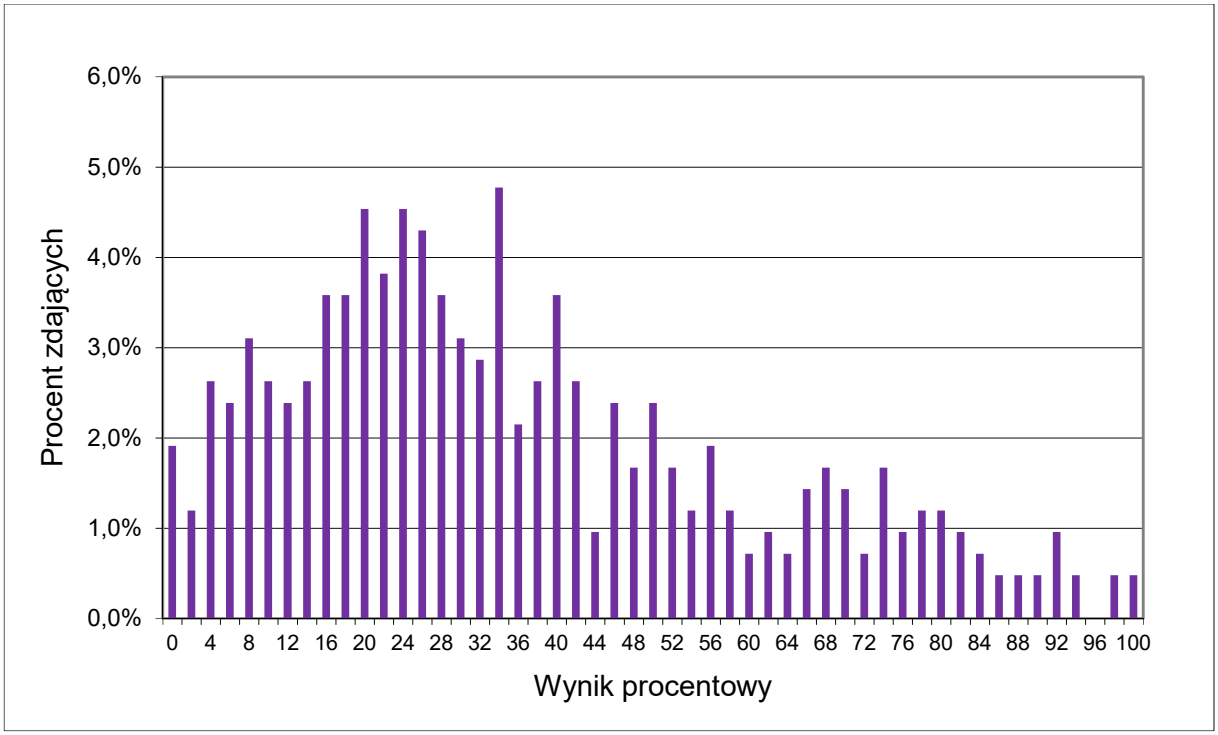



TABELA 4. WYNIKI ZDAJĄCYCH – PARAMETRY STATYSTYCZNE*

| Zdający | Liczba zdających | Minimum (%) | Maksimum (%) | Mediana (%) | Modalna (%) | Średnia (%) | Odchylenie standardowe (%) |
|---|---|---|---|---|---|---|---|
| **ogółem** | **419** | **0** | **100** | **32** | **34** | **36** | **24** |
| w tym: | | | | | | | |
| z liceów ogólnokształcących | 196 | 0 | 100 | 38 | 24 | 42 | 25 |
| z techników | 223 | 0 | 100 | 26 | 20 | 31 | 21 |

* Dane dotyczą wszystkich tegorocznych absolwentów. Parametry statystyczne są podane dla grup liczących 30 lub więcej zdających.

## Poziom wykonania zadań

TABELA 5. POZIOM WYKONANIA ZADAŃ

| Wymagania podstawy programowej | | | |
|---|---|---|---|
| **Nr zad.** | **Wymagania ogólne** | **Wymagania szczegółowe**<br>*Gdy wymaganie dotyczy treści zakresu podstawowego szkoły ponadpodstawowej – dopisano (P).* | **Poziom wykonania zadania (%)** |
| 1.1. | I. Rozumienie, analizowanie i rozwiązywanie problemów. | Zdający:<br>I.4) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych.<br>I+II.3) objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:<br>b) rekurencję.<br>P.I.3) sprawdza poprawność działania algorytmów dla przykładowych danych. | 65 |
| 1.2. | I. Rozumienie, analizowanie i rozwiązywanie problemów.<br>II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. | Zdający:<br>I.4) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych.<br>I+II.3) objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:<br>b) rekurencję.<br>P.I.3) sprawdza poprawność działania algorytmów dla przykładowych danych.<br>P.II.1) projektuje i programuje rozwiązania problemów z różnych dziedzin […]. | 37 |
| 1.3. | I. Rozumienie, analizowanie i rozwiązywanie problemów.<br>II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. | Zdający:<br>I.4) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych.<br>I+II.3) objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:<br>b) rekurencję.<br>P.I.3) sprawdza poprawność działania algorytmów dla przykładowych danych.<br>P.II.1) projektuje i programuje rozwiązania problemów z różnych dziedzin […]. | 45 |
| 2.1. | I. Rozumienie, analizowanie i rozwiązywanie problemów.<br>II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. | Zdający:<br>P.I.3) sprawdza poprawność działania algorytmów dla przykładowych danych.<br>P.II.1) projektuje i programuje rozwiązania problemów z różnych dziedzin […]. | 89 |

| | | | |
|---|---|---|---|
| 2.2. | I. Rozumienie, analizowanie i rozwiązywanie problemów.<br>II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. | Zdający:<br>I.4) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych.<br>I+II.3) objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów […].<br>P.I.3) sprawdza poprawność działania algorytmów dla przykładowych danych.<br>P.II.1) projektuje i programuje rozwiązania problemów z różnych dziedzin […]. | 38 |
| 3.1. | I. Rozumienie, analizowanie i rozwiązywanie problemów.<br>II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. | Zdający:<br>I.4) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych;<br>I.5) objaśnia dobrany algorytm, uzasadnia poprawność rozwiązania na wybranych przykładach danych i ocenia jego efektywność;<br>I.7) przedstawia sposoby reprezentowania w komputerze znaków, liczb […].<br>I+II.1) zapisuje za pomocą listy kroków, schematu blokowego lub pseudokodu, i implementuje w wybranym języku programowania, algorytmy poznane na wcześniejszych etapach […].<br>P.I.2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:<br>b) na tekstach […].<br>P.II.1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów [...]. | 35 |
| 3.2. | I. Rozumienie, analizowanie i rozwiązywanie problemów.<br>II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. | Zdający:<br>I.4) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych;<br>I.5) objaśnia dobrany algorytm, uzasadnia poprawność rozwiązania na wybranych przykładach danych i ocenia jego efektywność;<br>I.7) przedstawia sposoby reprezentowania w komputerze znaków, liczb […].<br>I+II.1) zapisuje za pomocą listy kroków, schematu blokowego lub pseudokodu, i implementuje w wybranym języku programowania, algorytmy poznane na wcześniejszych etapach […].<br>P.I.2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:<br>b) na tekstach […].<br>P.II.1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów [...]. | 28 |

| | | | |
|---|---|---|---|
| 3.3. | I. Rozumienie, analizowanie i rozwiązywanie problemów.<br>II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. | Zdający:<br>I.4) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych;<br>I.5) objaśnia dobrany algorytm, uzasadnia poprawność rozwiązania na wybranych przykładach danych i ocenia jego efektywność;<br>I.7) przedstawia sposoby reprezentowania w komputerze znaków, liczb […].<br>I+II.1) zapisuje za pomocą listy kroków, schematu blokowego lub pseudokodu, i implementuje w wybranym języku programowania, algorytmy poznane na wcześniejszych etapach […].<br>P.I.2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:<br>b) na tekstach […].<br>P.II.1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów [...]. | 10 |
| 4.1. | I. Rozumienie, analizowanie i rozwiązywanie problemów.<br>II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. | Zdający:<br>P.I.3) sprawdza poprawność działania algorytmów dla przykładowych danych.<br>P.I.2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:<br>a) na liczbach [...].<br>P.II.1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosując: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów [...]. | 85 |
| 4.2. | I. Rozumienie, analizowanie i rozwiązywanie problemów.<br>II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. | Zdający:<br>I.4) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych.<br>II.2) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów.<br>P.I.2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:<br>a) na liczbach [...].<br>P.II.1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów [...]. | 23 |
| 4.3. | I. Rozumienie, analizowanie i rozwiązywanie problemów.<br>II. Programowanie i rozwiązywanie problemów | Zdający:<br>I.4) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych. | 22 |

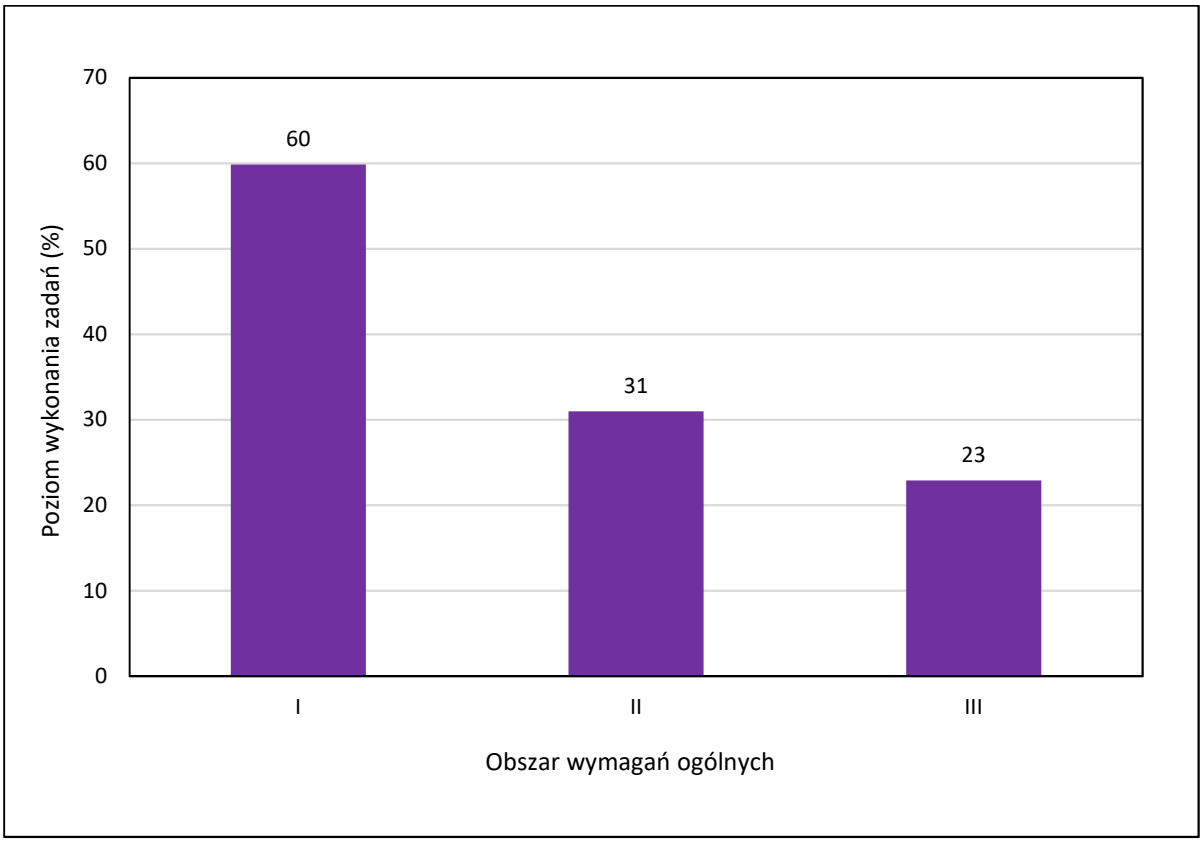
| | | | |
|---|---|---|---|
| | z wykorzystaniem komputera i innych urządzeń cyfrowych. | II.2) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów.<br>P.I.2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:<br>a) na liczbach [...].<br>P.II.1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów [...]. | |
| 4.4. | I. Rozumienie, analizowanie i rozwiązywanie problemów.<br>II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. | Zdający:<br>I.4) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych.<br>II.2) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów.<br>P.I.2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:<br>a) na liczbach [...].<br>P.II.1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów [...]. | 9 |
| 5. | I. Rozumienie, analizowanie i rozwiązywanie problemów.<br>II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. | Zdający:<br>P.I.2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy: a) na liczbach: […] zamiany reprezentacji liczb między pozycyjnymi systemami liczbowymi […].<br>I+II.2) wykorzystuje znane sobie algorytmy przy rozwiązywaniu i programowaniu rozwiązań następujących problemów:<br>b) wykonywania działań na liczbach w systemach innych niż dziesiętny. | 48 |
| 6. | III. Posługiwanie się komputerem, urządzeniami cyfrowymi i sieciami komputerowymi, w tym: znajomość zasad działania urządzeń cyfrowych i sieci komputerowych oraz wykonywania obliczeń i programów. | Zdający:<br>P.1) zapoznaje się z możliwościami nowych urządzeń cyfrowych i towarzyszącego im oprogramowania;<br>P.2) objaśnia funkcje innych niż komputer urządzeń cyfrowych i korzysta z ich Możliwości. | 23 |
| 7.1. | II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. | Zdający:<br>II.3) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym: | 77 |

| | | | |
|---|---|---|---|
| | | b) stosuje zaawansowane funkcje arkusza kalkulacyjnego w zależności od rodzaju danych.<br>P.II.3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami:<br>c) gromadzi dane pochodzące z różnych źródeł w tabeli arkusza kalkulacyjnego, korzysta z różnorodnych funkcji arkusza w zależności od rodzaju danych, filtruje dane według kilku kryteriów, dobiera odpowiednie wykresy do zaprezentowania danych, analizuje dane, korzystając z dodatkowych narzędzi, w tym z tabel i wykresów przestawnych. | |
| 7.2. | II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. | Zdający:<br>II.3) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym:<br>b) stosuje zaawansowane funkcje arkusza kalkulacyjnego w zależności od rodzaju danych.<br>P.II.3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami:<br>c) gromadzi dane pochodzące z różnych źródeł w tabeli arkusza kalkulacyjnego, korzysta z różnorodnych funkcji arkusza w zależności od rodzaju danych, filtruje dane według kilku kryteriów, dobiera odpowiednie wykresy do zaprezentowania danych, analizuje dane, korzystając z dodatkowych narzędzi, w tym z tabel i wykresów przestawnych. | 46 |
| 7.3. | II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. | Zdający:<br>II.3) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym:<br>b) stosuje zaawansowane funkcje arkusza kalkulacyjnego w zależności od rodzaju danych.<br>P.II.3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami:<br>c) gromadzi dane pochodzące z różnych źródeł w tabeli arkusza kalkulacyjnego, korzysta z różnorodnych funkcji arkusza w zależności od rodzaju danych, filtruje dane według kilku kryteriów, dobiera odpowiednie wykresy do zaprezentowania danych, analizuje dane, korzystając z dodatkowych narzędzi, w tym z tabel i wykresów przestawnych. | 23 |
| 7.4. | II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. | Zdający:<br>II.3) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym:<br>b) stosuje zaawansowane funkcje arkusza kalkulacyjnego w zależności od rodzaju danych.<br>P.II.3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami:<br>c) gromadzi dane pochodzące z różnych źródeł w tabeli | 22 |

| | | | |
|---|---|---|---|
| | | arkusza kalkulacyjnego, korzysta z różnorodnych funkcji arkusza w zależności od rodzaju danych, filtruje dane według kilku kryteriów, dobiera odpowiednie wykresy do zaprezentowania danych, analizuje dane, korzystając z dodatkowych narzędzi, w tym z tabel i wykresów przestawnych. | |
| 8.1. | II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. | Zdający:<br>II.3) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym:<br>c) projektuje i tworzy relacyjną bazę złożoną z wielu tabel, formułuje kwerendy, stosuje język SQL do wyszukiwania informacji w bazie i do jej modyfikacji, uwzględnia kwestie integralności danych, bezpieczeństwa i ochrony danych w bazie.<br>P.II.3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami:<br>d) wyszukuje informacje, korzystając z bazy danych opartej na co najmniej dwóch tabelach, definiuje relacje, stosuje filtrowanie, formułuje kwerendy. | 61 |
| 8.2. | II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. | Zdający:<br>II.3) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym:<br>c) projektuje i tworzy relacyjną bazę złożoną z wielu tabel, formułuje kwerendy, stosuje język SQL do wyszukiwania informacji w bazie i do jej modyfikacji, uwzględnia kwestie integralności danych, bezpieczeństwa i ochrony danych w bazie.<br>P.II.3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami:<br>d) wyszukuje informacje, korzystając z bazy danych opartej na co najmniej dwóch tabelach, definiuje relacje, stosuje filtrowanie, formułuje kwerendy. | 39 |
| 8.3. | II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. | Zdający:<br>II.3) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym:<br>c) projektuje i tworzy relacyjną bazę złożoną z wielu tabel, formułuje kwerendy, stosuje język SQL do wyszukiwania informacji w bazie i do jej modyfikacji, uwzględnia kwestie integralności danych, bezpieczeństwa i ochrony danych w bazie.<br>P.II.3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami:<br>d) wyszukuje informacje, korzystając z bazy danych opartej na co najmniej dwóch tabelach, definiuje relacje, stosuje filtrowanie, formułuje kwerendy. | 26 |
| 8.4. | II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. | Zdający:<br>II.3) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym:<br>c) projektuje i tworzy relacyjną bazę złożoną | 12 |

| | | | |
|---|---|---|---|
| | | z wielu tabel, formułuje kwerendy, stosuje język SQL do wyszukiwania informacji w bazie i do jej modyfikacji, uwzględnia kwestie integralności danych, bezpieczeństwa i ochrony danych w bazie.<br>P.II.3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami:<br>d) wyszukuje informacje, korzystając z bazy danych opartej na co najmniej dwóch tabelach, definiuje relacje, stosuje filtrowanie, formułuje kwerendy. | |
| 8.5. | II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. | Zdający:<br>II.3) przygotowując opracowania rozwiązań złożonych problemów, posługuje się wybranymi aplikacjami w stopniu zaawansowanym:<br>c) projektuje i tworzy relacyjną bazę złożoną z wielu tabel, formułuje kwerendy, stosuje język SQL do wyszukiwania informacji w bazie i do jej modyfikacji, uwzględnia kwestie integralności danych, bezpieczeństwa i ochrony danych w bazie.<br>P.II.3) przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami:<br>d) wyszukuje informacje, korzystając z bazy danych opartej na co najmniej dwóch tabelach, definiuje relacje, stosuje filtrowanie, formułuje kwerendy. | 27 |

WYKRES 2. POZIOM WYKONANIA ZADAŃ W OBSZARZE WYMAGAŃ OGÓLNYCH



# Komentarz

W roku 2026 do egzaminu maturalnego z informatyki w Formule 2023 przystąpili absolwenci liceów ogólnokształcących i techników. Egzamin odbył się tylko na poziomie rozszerzonym. Średni wynik, jaki osiągnęli wszyscy absolwenci (liceów oraz techników łącznie), wynosi **40%**. Absolwenci liceów osiągnęli średni wynik **46%**, natomiast absolwenci techników osiągnęli średni wynik **32%**.

### Analiza jakościowa zadań

Arkusz składał się z 23 zadań. 21 zadań ujęto w 6 wiązek tematycznych, a dwa zadania były odrębne. Analiza poziomu wykonania zadań zamieszczona w Tabeli 5. pokazuje, że dla tegorocznych maturzystów 3 zadania były bardzo trudne (poziom wykonania do 19%), 14 zadań było trudnych (20–49%), 2 zadania były umiarkowanie trudne (50–69%), 3 zadania były łatwe (70–89%), a 1 zadanie było bardzo łatwe (powyżej 89%).

### Poziomy trudności zadań

Rozkład punktacji na poszczególnych poziomach trudności:
• za zadania bardzo trudne (poziom wykonania w zakresie od 0%-19%) można było uzyskać łącznie 9 punktów, czyli 18% maksymalnej liczby punktów do zdobycia
• za zadania trudne (poziom wykonania w zakresie od 20%-49%) – 30 punktów (60%)
• za zadania umiarkowanie trudne (poziom wykonania w zakresie od 50%-69%) – 3 punkty (6%)
• za zadania łatwe (poziom wykonania w zakresie od 70%-89%) – 7 punktów (14%)
• za zadanie bardzo łatwe – 1 punkt (2%).
W tym roku zadania były bardziej zróżnicowane pod względem łatwości w stosunku do roku 2025, w którym nie pojawiło się ani jedno zadanie bardzo łatwe, ani też bardzo trudne dla zdających.

W tegorocznym arkuszu nie znalazło się zadanie zamknięte. Arkusz złożony był z zadań otwartych, przeznaczonych do wykonania w arkuszu, oraz z zadań praktycznych. Zadania praktyczne wymagają przestawienia komputerowej realizacji rozwiązania. W arkuszu takimi zadaniami są: 3.1.–3.3., 4.2.–4.4., 7.1.–7.4. oraz 8.1.–8.4. W tych zadaniach zapis samego wyniku nie zastępował wymaganych plików z komputerową realizacją zadania. Brak plików zawierających komputerową realizację zadań był jednoznaczny z brakiem rozwiązania tych zadań.
Za rozwiązanie zadań praktycznych można było uzyskać łącznie 32 punkty, czyli 64% wszystkich możliwych do zdobycia.

### Zadania, z którymi zdający poradzili sobie najsłabiej

Do pogłębionej analizy przyjęto zadania, których poziom wykonania był niższy niż 35%. Takich zadań w arkuszu było 11, w tym 3, które okazały się dla zdających bardzo trudne. Zadania te, w kolejności od najtrudniejszego, to:

- Zadanie 4.4. (12%, programistyczne, praktyczne – operacje na tablicy, hierarchia pracowników)
- Zadanie 3.3. (14%, programistyczne, praktyczne – operacje na napisach)
- Zadanie 8.4. (14%, bazodanowe, praktyczne)
- Zadanie 6. (22%, wiedza o sieciach komputerowych, otwarte)
- Zadanie 4.2. (28%, programistyczne, praktyczne)
- Zadanie 4.3. (28%, programistyczne, praktyczne)
- Zadanie 8.5. (29%, bazodanowe – zapis zapytania SQL w arkuszu)
- Zadanie 3.2. (31%, programistyczne, praktyczne – operacje na napisach, zliczanie liter)
- Zadanie 7.4. (31%, praktyczne – symulacja)
- Zadanie 7.3. (32%, praktyczne – symulacja)
- Zadanie 8.3. (32%, bazodanowe, praktyczne).

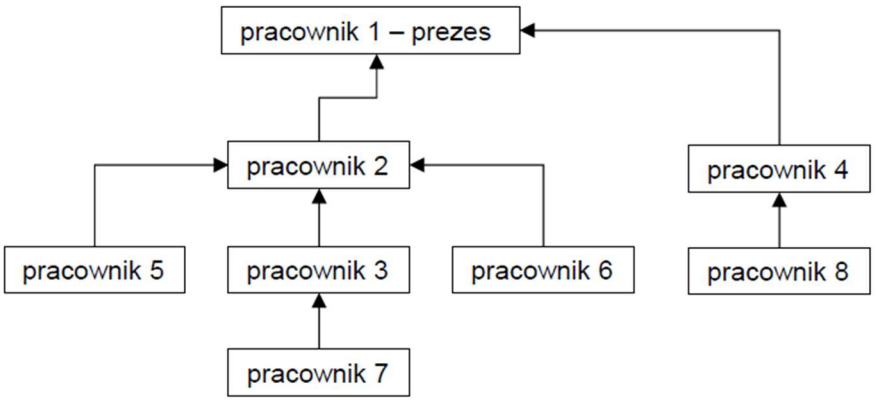
## Zadania 4.2.–4.4. – „Korporacja”

Najtrudniejszym zadaniem w całym arkuszu było zadanie 4.4. należące do wiązki zadań opartej na hierarchii pracowników w fikcyjnej korporacji. Dane do zadania określały bezpośrednich przełożonych danego pracownika.
Podstawowe założenia przedstawiono na początku zadania (Rysunek 1.). Dla łatwiejszego zrozumienia treści podano przykład ilustrowany rysunkiem. Do dyspozycji zdających był także przykładowy plik, na którym można było sprawdzać działanie swoich programów.

**Zadanie 4. Korporacja**

W korporacji pracuje *n* osób, które na potrzeby zadania ponumerujemy liczbami 1, 2, …, *n*. Pracownik numer 1 jest prezesem korporacji, a każdy z pozostałych pracowników ma dokładnie jednego bezpośredniego przełożonego.
Numer bezpośredniego przełożonego pracownika *x* jest zawsze mniejszy od numeru tego pracownika.
Prezes korporacji nie ma żadnego przełożonego. Przełożonym pracownika jest jego bezpośredni przełożony i każdy przełożony tego bezpośredniego przełożonego.
Jeśli *x* jest (bezpośrednim) przełożonym *y*, to powiemy, że *y* jest (bezpośrednim) podwładnym *x*. Prezes jest przełożonym każdego pracownika.

**Przykład 1.**




*Rysunek 1.* Przykład hierarchii w korporacji (strzałki wskazują na bezpośrednich przełożonych).
Przełożonymi pracownika 7 są pracownicy 3, 2 i 1. Bezpośrednim przełożonym pracownika 7 jest pracownik 3. Pracownik 3 (podobnie jak 5 oraz 6) jest bezpośrednim podwładnym pracownika 2, a podwładnymi pracownika 2 są pracownicy 3, 5, 6 i 7.

**Rysunek 1.**

Plik opisujący hierarchię pracowników zawierał liczby oznaczające numery bezpośrednich przełożonych pracowników o numerach takich jak numery wierszy. Tj. w wierszu pierwszym zapisano 0 oznaczające brak przełożonego nad prezesem korporacji – pracownikiem numer 1, w wierszu drugim zapisany był numer bezpośredniego przełożonego pracownika numer 2, w trzecim – numer bezpośredniego przełożonego pracownika numer 3 itd. Zgodnie z treścią zadania numer przełożonego był zawsze mniejszy niż numer jego podwładnego – czyli im mniejszy numer tym wyższa pozycja w korporacji.

W zadaniu 4.2. (poziom wykonania 28%, Rysunek 2.) należało podać liczbę pracowników, którzy nie byli przełożonymi żadnego innego pracownika. Biorąc pod uwagę strukturę danych należało zauważyć, że skoro numery przełożonych są zapisane w pliku, to wystarczyło zliczyć ile numerów nie jest w tym pliku zapisanych (liczba wierszy pliku zgadzała się z liczbą wszystkich pracowników). Czyli można było wykonać „słownik” – tablicę, w której odznaczamy, czy pod danym numerem jest czyjś przełożony, czy też nie, albo dla każdego numeru od 1 do liczby pracowników sprawdzić, czy ten numer jest w pliku.

**Zadanie 4.2. (0–2)**
Na podstawie danych zapisanych w pliku `korpo.txt` podaj, ilu pracowników nie jest przełożonym żadnego pracownika.

Dla danych z pliku `korpo_przyklad.txt` poprawna odpowiedź to:
49 998

**Rysunek 2.**

Zadanie 4.2. – przykładowe rozwiązanie zdającego w języku c++:

```
const int N = 50000;
[…]
bool zlicz[N + 1] = {false};
int wynik = 0;
for(int i = 1; i <= N; i++)
{
    zlicz[dane[i]]= true;
}
for(int i = 1; i <= N; i++)
{
    if(zlicz[i] == false) wynik++;
}
wypisz<<"Zadanie 4.2"<<endl;
wypisz<<wynik<<endl<<endl;
```

Zdający zadeklarował tablicę zawierającą wartości logiczne i najpierw ustawił wartości wszystkich jej elementów na fałsz. Następnie dla każdego indeksu, który reprezentuje czyjegoś przełożonego zmieniał wartość na prawda. W kolejnej pętli zliczył te elementy, które pozostały z wartością fałsz, nie zapomniał także o pominięciu przy zliczaniu indeksu 0 tablicy – nie ma w zadaniu pracownika o numerze 0.

W zadaniu 4.3. (poziom wykonania 28%, Rysunek 3.) należało policzyć, który pracownik ma najwięcej bezpośrednich podwładnych. Biorąc pod uwagę, że w pliku zapisano bezpośrednich przełożonych pracowników o numerach zgodnych z numerami wierszy, to wystarczyło zliczyć dla każdego z nich, ile razy występuje w pliku. A więc zadanie to jest klasycznym zadaniem na wyszukanie najczęściej występującego elementu w podanym zestawie liczb.

**Zadanie 4.3. (0–2)**
Podaj, który pracownik spośród zapisanych w pliku `korpo.txt` ma najwięcej bezpośrednich podwładnych. Podaj numer tego pracownika oraz liczbę jego bezpośrednich podwładnych.

Dla danych z pliku `korpo_przyklad.txt` poprawna odpowiedź to:
3 49 997
(pracownik numer 3 ma 49 997 bezpośrednich podwładnych).

**Rysunek 3.**

Zadanie 4.3. – przykładowe rozwiązanie zdającego w języku c++:

```
int zlicz[N + 1] = {0};
int max_wynik = 0, odp;
for(int i = 1; i <= N; i++)
{
    zlicz[dane[i]]++;
}
for(int i = 1; i <= N; i++)
{
    if(zlicz[i] > max_wynik)
    {
        max_wynik = zlicz[i];
        odp = i;
    }
}
wypisz<<"Zadanie 4.3"<<endl;
wypisz<<odp<<" "<<max_wynik<<endl<<endl;}
```

Zdający wykorzystał klasyczną metodę zliczania wystąpień elementu. W pierwszej pętli zwiększa o 1 wartość w tablicy *zlicz* na indeksie reprezentującym numer przełożonego. Następnie, w drugiej pętli wyszukuje maksymalną wartość w tablicy *zlicz* i zapisuje indeks reprezentujący numer przełożonego mającego najwięcej podwładnych.

Zadanie 4.4. (poziom wykonania 12%, Rysunek 4.) było najtrudniejszym zadaniem w całym arkuszu. W zadaniu tym należało policzyć dla każdego pracownika ilu przełożonych (od bezpośredniego przełożonego aż do prezesa) jest „nad nim”. Następnie należało podać ilu maksymalnie przełożonych może mieć pracownik oraz ilu jest pracowników, którzy mają tylu przełożonych.

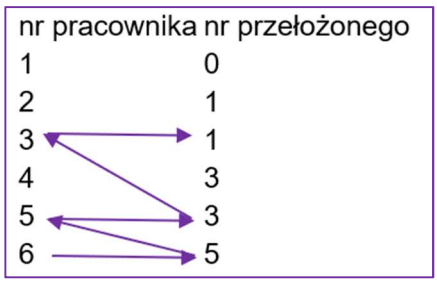
**Zadanie 4.4. (0–3)**
Policz, ilu najwięcej przełożonych ma jeden pracownik. Podaj tę liczbę oraz podaj, ilu pracowników ma taką liczbę przełożonych.

Dla danych z pliku `korpo_przyklad.txt` poprawna odpowiedź to:
2 49 997
(największa liczba przełożonych: 2, liczba pracowników, którzy mają 2 przełożonych: 49 997).

**Rysunek 4.**

Tylko pozornie jest to trudne do policzenia. Najprostszym podejściem było zauważenie, że aby policzyć liczbę przełożonych danego pracownika musimy poruszać się w tablicy w stronę jej początku „przechodząc” po kolejnych przełożonych aktualnego przełożonego, aż dojdziemy do 1, czyli do prezesa.

**Przykład:**




**Rysunek 5.**

W przykładzie pokazanym na Rysunku 5. pracownik numer 6 ma nad sobą 3 przełożonych: pracownika 5 – bezpośredniego przełożonego, pracownika 3 – przełożonego pracownika 5, oraz prezesa, który jest jedynym przełożonym pracownika 3.
Przykładowy zapis takiego podejścia może wyglądać jak na listingu poniżej (zapis w c++):

```
ifstream p("korpo.txt");
int przelozony[50000];
for (int i=0;i<50000;i++)
    {   int x;
        p>>x;
        przelozony[i]=x-1;
    }
int glebokosc[50000];
int max=0;
int ile=0;
for (int i=1;i<50000;i++)
    {   int glebokosc=1;
        int pracownik=i;
        while(przelozony[pracownik]!=0)
        {   glebokosc++;
            pracownik=przelozony[pracownik];
        }
        if (glebokosc>max)
            {max=glebokosc;
             ile=0;}
        if (glebokosc==max)
            ile++;
     }
cout<<max<<" "<<ile<<endl;
```

Bardziej eleganckie rozwiązanie, mające złożoność liniową względem liczby pracowników, bazuje na kluczowej obserwacji: jeżeli przełożony[*i*] jest bezpośrednim przełożonym pracownika *i*, to liczba wszystkich przełożonych pracownika *i* jest o 1 większa od liczby przełożonych pracownika przełożony[*i*]. Ponieważ przełożony[*i*] < *i*, potrzebna wartość była już wcześniej obliczona.
Przykładowe rozwiązanie dla drugiego podejścia (fragment, zapis w pseudokodzie):

```
glebokosc[1] = 0
max = 0
ile = 1
dla i = 2..n:
    glebokosc[i] = glebokosc[p[i]] + 1
    jeżeli glebokosc[i] > max:
        max = glebokosc[i]
        ile = 1
    w przeciwnym razie
        jeżeli glebokosc[i] = max:
           ile = ile + 1
wypisz max, ile
```

## Zadanie 4.4. – przykładowe rozwiązania zdających

**Rozwiązanie 1.**
Poniższy fragment kodu zawiera poprawne rozwiązanie zadania 4.4. w języku Python:

```
plik = open("korpo.txt",'r')
l = [0]*50001
for i in range(1,50001):
    n = int(plik.readline())
    l[i] = n

maksprzelozonych = 0
ilemamaks = 0
for i in range(50001):
    currprzelozony = l[i]
    ileprzelozonych = 0
    while currprzelozony != 0:
        currprzelozony = l[currprzelozony]
        ileprzelozonych+=1
    if ileprzelozonych > maksprzelozonych:
        ilemamaks = 0
        maksprzelozonych = ileprzelozonych
    if ileprzelozonych == maksprzelozonych:
        ilemamaks+=1
print(maksprzelozonych, ilemamaks)
```

Zdający poprawnie wczytuje numery przełożonych do tablicy *l* , tak aby indeks odpowiadał numerowi pracownika (zaczyna od 1). Następnie dla każdego pracownika (w pętli while) przechodzi po kolejnych przełożonych aż do prezesa, prawidłowo zliczając liczbę wszystkich przełożonych. Poprawnie także liczone jest maksimum liczby przełożonych i liczba pracowników.
Jak widać, jest to pierwszy z omawianych sposobów rozwiązania zadania.

**Rozwiązanie 2.**
Kolejny przykład  to fragment kodu zawierający poprawne rozwiązanie w języku c++:

```
    fstream plik;
    plik.open("korpo.txt");
    int jakiprzelozony[50001];
    int iluprzelozonych[50001];
    int maxx_przelozonych = 0;
    int maxx_przelozonych_ile = 0;

    for(int i=1; i<=50000; i++)
```

```
    {
        plik >> jakiprzelozony[i];
        ilupodwladnych[i] = 0;
        iluprzelozonych[i] = 0;
    }
[…]

    iluprzelozonych[1] = 0;
    for(int i=2; i<=50000; i++)
    {
        iluprzelozonych[i]=1+iluprzelozonych[jakiprzelozony[i]];
        if(iluprzelozonych[i]>maxx_przelozonych)
        {
            maxx_przelozonych = iluprzelozonych[i];
        }
    }

[…]
    cout << "4.4" << endl;
    cout << maxx_przelozonych << endl;
    for(int i=1; i<=50000; i++)
    {
        if(iluprzelozonych[i]==maxx_przelozonych)
        {
            maxx_przelozonych_ile++;
        }
    }
    cout << maxx_przelozonych_ile << endl;
```

Zdający skorzystał z drugiego z wcześniej omawianych podejść do rozwiązania zadania wykorzystującego fakt, że liczba przełożonych pracownika x jest o 1 większa od liczby przełożonych bezpośredniego przełożonego pracownika x.

**Rozwiązanie 3. Rozwiązanie z błędem (kod w c++).**

```
int N,n;
int i,x;
int a;
int t[500001];
int t2[500001];
int ile;
int maxi=0;

int p;

for(i=1;i<=n;i++)
    {
        plo>>t[i];
        t2[t[i]]++;
    }
for(i=2;i<=n;i++)
      {
       p=1;
       x=t[i];
       while (t[x]!=0)
       {
           p++;
           x=t[x];
```

```
        }
        if(p>maxi)
        {
            maxi=p;
            ile=1;
        }
        if(p==maxi)
         ile++;
    }
cout<<maxi<<" "<<ile;
}
```

Kolejne rozwiązanie zawiera jeden błąd powodujący nieprawidłowe zliczanie liczby pracowników. Samo liczenie liczby przełożonych jest wykonane poprawnie. Gdy znalezione zostanie nowe maksimum liczby przełożonych, program najpierw podstawia 1 do zmiennej *ile* zliczającej liczbę pracowników, a następnie zwiększa tą zmienną o 1 – ponieważ kolejny warunek także jest spełniony. W rezultacie *ile* rośnie do 2, mimo że znaleziono dopiero jednego pracownika z takim maksimum. Po pierwszym warunku (*p>maxi*) należało zastosować *else*. W rezultacie zdający otrzymuje 2 zamiast 3 punktów za to zadanie (za poprawną liczbę przełożonych).
Ten błąd powinien zostać wykryty już dla danych przykładowych. Dla pliku `korpo_przyklad.txt` podano w arkuszu prawidłową odpowiedź 2 49997, program dla tego pliku daje wynik 2 49998.
Zadanie 4.4. było bardzo często pomijane przez zdających.

### Zadania 3.2. i 3.3. – „Pary słów”

Drugie z najtrudniejszych zadań w arkuszu – zadanie 3.3. o poziomie wykonania 14%, należało do wiązki zadań „Pary słów” (Rysunek 6.). Wiązka oparta była o dane w pliku zawierającym pary napisów. Wszystkie zadania wiązki wymagały umiejętności operacji na łańcuchach znaków. Pozostałe dwa zadania z wiązki 3.1. i 3.2. (poziom wykonania 39% i 31%) także należały do trudnych dla zdających. Ponieważ założyliśmy, że pochylimy się nad zadaniami o poziomie wykonania poniżej 35%, pominiemy opis zadania 3.1.

**Zadanie 3. Pary słów**

W pliku tekstowym `pary.txt` znajduje się 500 par słów złożonych z liter alfabetu angielskiego *a*, *b*, …, *z*. Każda para słów jest zapisana w osobnym wierszu. Słowa w wierszu są oddzielone pojedynczym odstępem, a długość każdego z nich nie przekracza 50 znaków.

Pierwszych pięć wierszy pliku `pary.txt` zawiera następujące pary słów:

```
bcba babb
abaa ccc
bcb abbba
bca cdd
aadc ddcdccaba
```

**Napisz program** (lub kilka programów), który(-e) znajdzie(-ą) i da(dzą) odpowiedzi do podanych zadań. Odpowiedzi do poszczególnych zadań zapisz w pliku `wyniki3.txt`. Każdą odpowiedź poprzedź numerem oznaczającym zadanie.

**Rysunek 6.**

W zadaniu 3.2. (Rysunek 7.) należało dla każdej litery alfabetu policzyć jej wystąpienia w obu słowach, a następnie zsumować minima tych liczności. Naturalnym rozwiązaniem było

użycie dwóch tablic 26-elementowych. Zadanie sprawdzało więc nie tylko iterowanie po napisach, lecz także dobór prostej struktury danych do zliczania.

**Zadanie 3.2. (0–3)**

Wspólną liczbę wystąpień litery $x$ w słowach $s_1$, $s_2$ oznaczymy przez $W(x, s_1, s_2)$ i definiujemy jako

$$W(x, s_1, s_2) = \text{minimum}(d(x, s_1), d(x, s_2))$$

gdzie $d(x, s)$ oznacza liczbę wystąpień litery $x$ w słowie $s$.

Podaj parę słów występujących w jednym wierszu w pliku `pary.txt`, dla której suma wspólnych wystąpień wszystkich liter jest największa, oraz podaj tę sumę. Jest jedna taka para.

**Rysunek 7.**

Przykładowe poprawne rozwiązanie zdającego w języku Python:

```
def W(x,s1,s2):
    return min(s1.count(x),s2.count(x))
with open("pary.txt") as file:
    dane=file.readlines()
dane=list(map(lambda x: x.split(),dane))
print(dane)
maxi=0
para_max=[]
for x in dane:
    l=0
    for y in set("".join([x[0],x[1]])):
        l+=W(y,x[0],x[1])
    if l>maxi:
        maxi=l
        para_max=x
print(para_max[0],para_max[1],maxi)
print()
```

Funkcja *W* jest przepisaniem definicji z zadania. Ciekawym zabiegiem jest użycie zbioru – *set* w celu uniknięcia zliczania wielokrotnie tych samych liter.

Przykładowe poprawne rozwiązanie zdającego w języku c++:

```
int x(string a, string b) {
	int la[26] = {0};
	int lb[26] = {0};
	for (int i = 0; i < a.size(); i++) la[a[i] - 'a']++;
	for (int i = 0; i < b.size(); i++) lb[b[i] - 'a']++;

	int sum = 0;
	for (int i = 0; i < 26; i++) {
		if (la[i] < lb[i]) sum += la[i];
		else
			sum += lb[i];
	}
	return sum;
}
int main() {
	ifstream dane("pary.txt");
	string a, b;
	int max1 = 0;
```

```
        string max_a, max_b;

        int max2 = 0;
        string s1, s2;

        for (int i = 0; i < 500; i++) {
                dane >> a >> b;
                if (x(a,b) > max2) {
                        max2 = x(a,b);
                        s1 = a;
                        s2 = b;
                }

        cout << << s1 << " " << s2 << " " << max2;
}
```

Funkcja *x* zlicza wspólną liczbę wystąpień liter w słowach *a* i *b*.

Zadanie 3.3. (poziom wykonania 14%, Rysunek 8.), drugie pod względem trudności zadanie w arkuszu, wymagało jednoczesnego uwzględnienia dwóch kierunków dopasowania: początku pierwszego słowa do końca drugiego oraz początku drugiego słowa do końca pierwszego. Dodatkową trudnością było znalezienie najdłuższego dopasowania, a następnie wybranie wszystkich par, dla których jego długość wynosiła co najmniej 5.

**Zadanie 3.3. (0–4)**

*Prefiksosufiksem* pary słów $s_1$, $s_2$ nazywamy słowo, które jest początkiem $s_1$ (czyli $s_1$ zaczyna się tym słowem) oraz końcem $s_2$ (czyli $s_2$ kończy się tym słowem) lub początkiem $s_2$ oraz końcem $s_1$.

Podaj wszystkie pary słów z pliku `pary.txt`, dla których **najdłuższy** prefiksosufiks ma **co najmniej 5** liter. Dla każdej podanej w odpowiedzi pary słów podaj długość najdłuższego prefiksosufiksu tej pary.

**Przykłady:**
Dla pary `aabbbca caacaab` mamy następujące prefiksosufiksy:

- `aab` – początek pierwszego słowa i koniec drugiego
- `ca` – początek drugiego słowa i koniec pierwszego.

Najdłuższy prefiksosufiks ma długość 3, zatem para tych słów nie spełnia wymaganych warunków.

Dla pary `abbaabaa baabaabba` mamy następujące prefiksosufiksy:

- `a` – początek pierwszego słowa i koniec drugiego
- `abba` – początek pierwszego słowa i koniec drugiego
- `baa` – początek drugiego słowa i koniec pierwszego
- `baabaa` – początek drugiego słowa i koniec pierwszego.

Najdłuższy prefiksosufiks ma długość 6, zatem para spełnia warunki wymagane w zadaniu.

Dla pliku `pary_przyklad.txt` poprawną odpowiedzią jest
ecedddeed dddeedd 6
(najdłuższy prefiksosufiks ma długość 6)

**Rysunek 8.**

Przy długości słów nieprzekraczającej 50 znaków można było bezpośrednio sprawdzić kolejne możliwe długości prefiksosufiksu:

```
dla każdej pary s1, s2:
    best = 0
    dla k = 1..min(dlugosc(s1), dlugosc(s2)):
        jeżeli prefix(s1,k) = suffix(s2,k) lub prefix(s2,k) = suffix(s1,k):
            best = k
    jeżeli best >= 5:
        wypisz s1, s2, best
```

To zadanie było często pomijane przez zdających.

### Zadanie 3.3. – przykładowe rozwiązania zdających

**Rozwiązanie 1. Fragment poprawnego kodu zdającego w języku c++.**

```
int sufix(string a, string b) {
	int dl;
	if (a.size() < b.size())
		dl = a.size();
	else
		dl = b.size();

	string sa = "", sb = "";
	int p = 0;
	for (int i = 0; i < dl; i++) {
		sa = sa + a[i];
		sb = b[b.size() - i - 1] + sb;
		if (sa == sb) p = sa.size();
	}
	return p;
}

int main() {
	ifstream dane("pary.txt");
	string a, b;

	int max1 = 0;
	string max_a, max_b;

	int max2 = 0;
	string s1, s2;

	for (int i = 0; i < 500; i++) {
	  dane >> a >> b;
	  if (sufix(a,b) >= 5 || sufix(b,a) >= 5)
	    cout << a << " " << b << " " << max(sufix(a,b),sufix(b,a)) << endl;
	}
}
```

Funkcja *sufix* wyszukuje najdłuższy prefiksosufiks tylko dla wersji, gdzie bierzemy początek pierwszego słowa i koniec drugiego. Jednak przy sprawdzaniu długości i wypisywaniu sprawdzane są obie wersje dzięki wywołaniu funkcji z parametrami a, b i b, a.

### Rozwiązanie 2. Poprawne rozwiązanie w języku Python

```
plik = open('pary.txt','r')
k = 500
L = []
for i in range(500):
    a, b = map(str, plik.readline().split())
    j = min(len(a), len(b))
    temp = 0
    p1 = '-1'
    p2 = '-2'
    for j in range(1,j+1):
        prefA = a[:j]
        suffA = a[len(a)-j:]
        prefB = b[:j]
        suffB = b[len(b)-j:]

        # prefA suff B
        if prefA == suffB and len(prefA)>=5 and temp < len(prefA):
            temp = len(prefA)
            p1 = a
            p2 = b
        if prefB == suffA and len(prefB) >= 5 and temp < len(prefB):
            temp = len(prefB)
            p1 = a
            p2 = b
    if p1 != '-1':
        L.append((p1,p2,temp))
for elem in L:
    print(*elem)
```

Tutaj zdający od razu sprawdza prefiksodufiksy w obie strony oraz czy maksymalna długość jest większa bądź równa 5.

### Rozwiązanie 3. Rozwiązanie w języku Python za 2 punkty

```
with open("./Dane/pary.txt") as file:
    content = file.readlines()
    pary = []
    for line in content:
        pary.append(line.split())
    print(pary)
def czy_duzy_presufix(a,b):
    max_presufix_front = 0
    max_presufix_back = 0
    n = len(a) if len(a) < len(b) else len(b)
    for i in range(n):
        front_a = a[:i]
        front_b = b[-i:]
        if a[:i] == b[-i:]:
            max_presufix_front = i
        back_a = a[-i:]
        back_b = b[:i]
        if b[:i] == a[-i:]:
            max_presufix_back = i
    return max(max_presufix_front, max_presufix_back)
```

```
print("3.3")
curr_max_presufix = 0
curr_para = None
for para in pary:
    max_presufix = czy_duzy_presufix(para[0],para[1])
    if max_presufix > curr_max_presufix:
        curr_max_presufix = max_presufix
        curr_para = para
print(f"{curr_para[0]}, {curr_para[1]}, {curr_max_presufix}\n\n")
```

Zdający poprawnie szuka największego prefikosufiksu dla pary napisów a i b w funkcji *czy_duzy_presufix*, jednak potem szuka w całym pliku najdłuższego prefiksosufiksu, podczas gdy należało tylko wypisać te najdłuższe prefisosufiksy par, które mają długość mniejszą bądź równą 5. Za takie rozwiązanie przewidziano w zasadach oceniania 2 punkty z 4.

### Zadania 8.3.–8.5. – „Sieć sklepów”

Równie trudne dla zdających jak zadanie 3.3. okazało się jedno z zadań w wiązce bazodanowej „Sieć sklepów” (Rysunek 9.) – zadanie 8.4 . Zadania z tej wiązki wymagały umiejętności łączenia informacji z kilku tabel, właściwego traktowania brakujących wartości oraz agregowania danych.

**Zadanie 8. Sieć sklepów**

W trzech plikach tekstowych o nazwach `klienci.txt`, `transakcje.txt`, `opis_transakcji.txt` zapisano dane o sprzedaży towarów w pewnej sieci sklepów w porach wieczornych pierwszych dni miesiąca. Dane obejmują informacje od 1. do 3. dnia miesiąca w miesiącach od stycznia do czerwca 2025 roku w godzinach od 22:00 do 23:59. Pierwszy wiersz każdego z plików jest wierszem nagłówkowym, a dane w wierszach rozdzielono tabulatorami.

**Rysunek 9.**

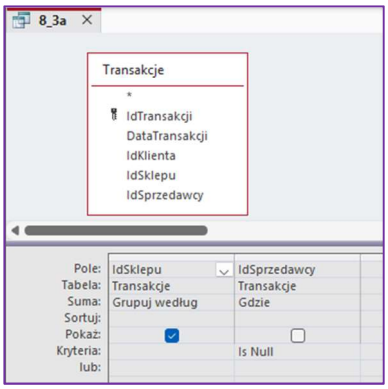
Do najtrudniejszych zadań z tej wiązki oprócz zadania 8.4. należały także 8.3. i 8.5. Zadanie 8.3. (32%, Rysunek 10.) łączyło filtr warunku „kasa samoobsługowa” z obliczeniem wartości zakupów.

**Zadanie 8.3. (0–2)**

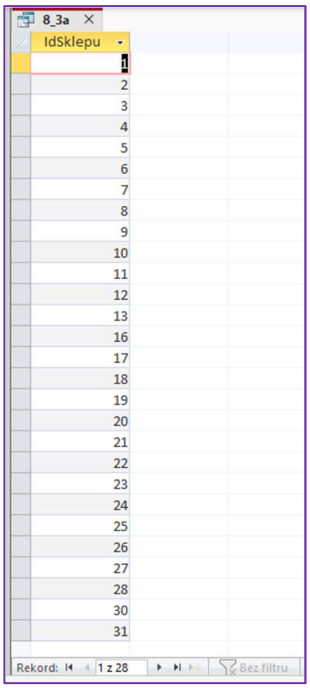
Podaj liczbę różnych sklepów, w których dokonano transakcji w kasach samoobsługowych, oraz podaj, ile zapłacono łącznie za zakupy w tych kasach.

**Rysunek 10.**

Zgodnie z opisem plików do zadania, dane transakcji w kasie samoobsługowej nie zawierały idSprzedawcy. Po wybraniu transakcji z pustym IdSprzedawcy należało połączyć je z opisem transakcji i sumować iloczyny Cena × Liczba. Trzeba było policzyć także liczbę różnych sklepów.
Przykładowe rozwiązanie w Access:
Wyszukiwanie różnych sklepów z transakcjami w kasach samoobsługowych (Rysunek 11.)

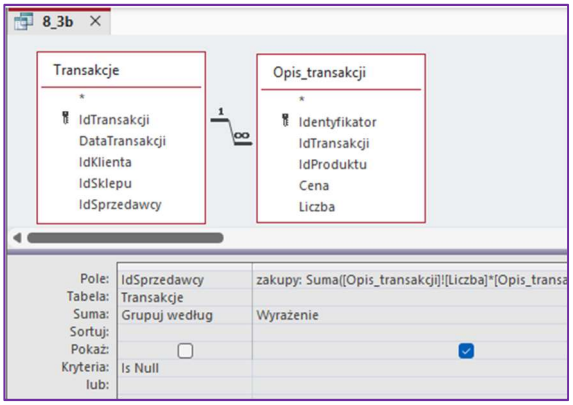



**Rysunek 11.**

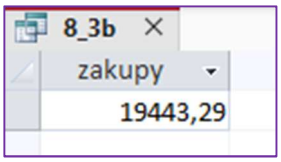



**Rysunek 12.**

Zliczanie, ile zapłacono za zakupy:

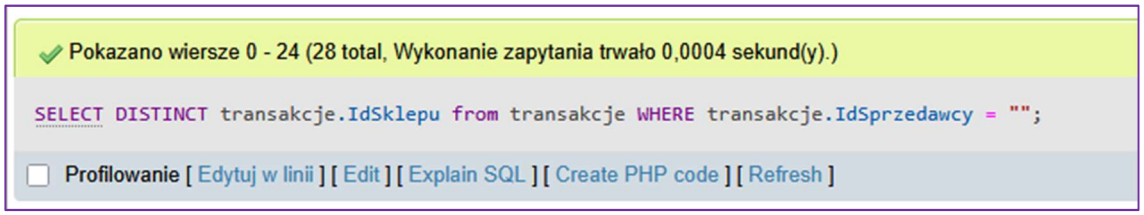



**Rysunek 13.**



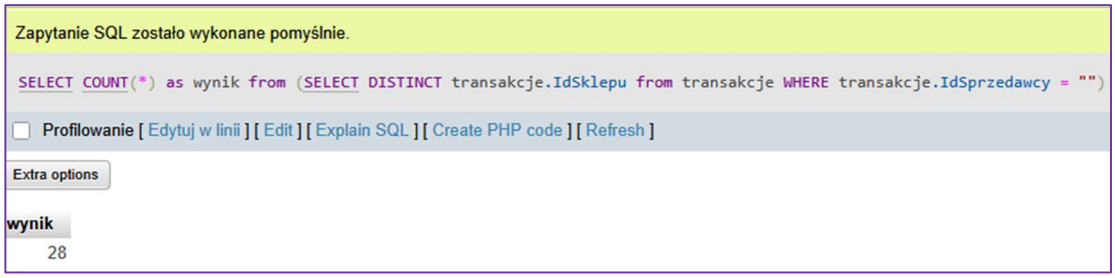

**Rysunek 14.**

Przykładowe rozwiązania w mySQL (phpMyAdmin):

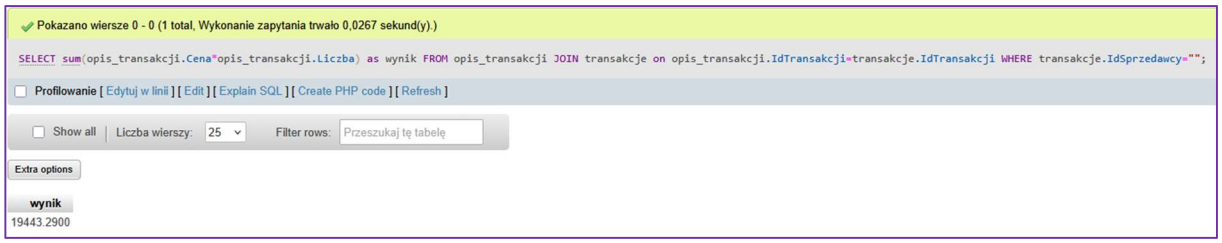



**Rysunek 15.**

Uwaga: powyższy zapis jest poprawny w przypadku, gdy przy imporcie kolumna IdSprzedawcy ustawiona została jako tekst, inaczej powinno być: IdSprzedawcy is NULL.



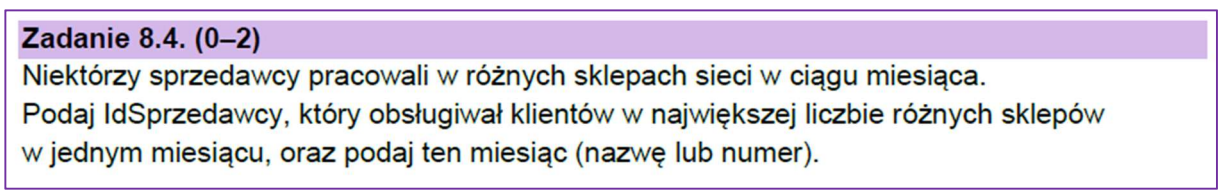

**Rysunek 16.**




**Rysunek 17. Zadanie 8.3 – ile wydano na zakupy.**

Uwaga: w przypadku rozwiązywania zadania z użyciem pakietu XAMPP z zainstalowanym MySQL, phpMyAdmin, jako komputerową realizację należało dołączyć wyeksportowaną bazę danych w formacie SQL. Nie wszyscy zdający o tym pamiętali. W przykładowym rozwiązaniu widać, że inaczej zbudowana baza powoduje też różnice w zapytaniach SQL (np. typ danych w danej kolumnie).

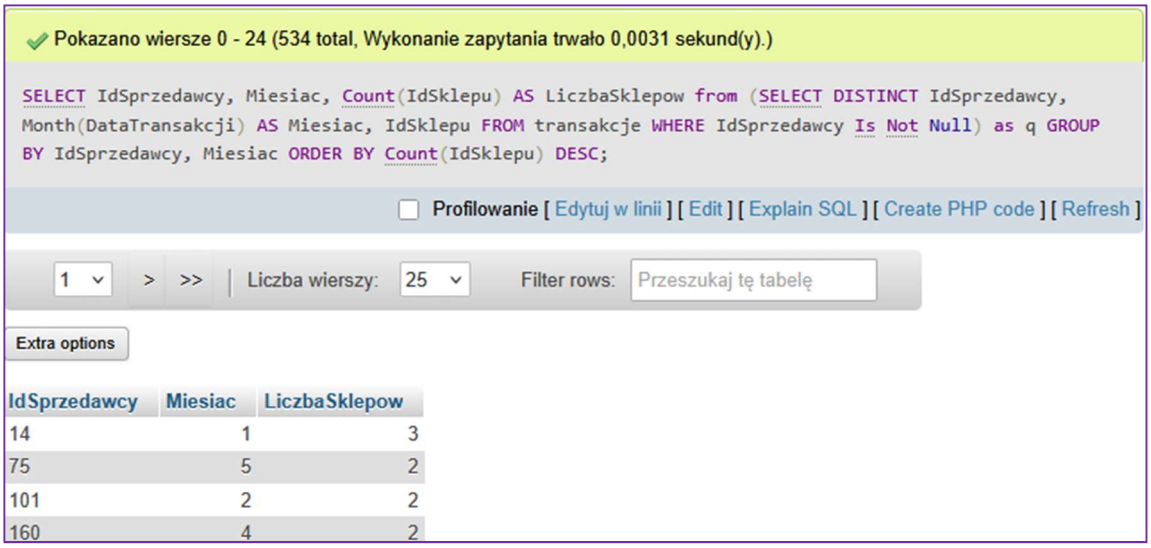
W wiązce bazodanowej najtrudniejszym dla zdających okazało się zadanie 8.4. (poziom wykonania 14%, Rysunek 18), w którym należało dla każdego sprzedawcy i miesiąca ustalić liczbę różnych sklepów, a następnie wskazać największą z tych wartości.



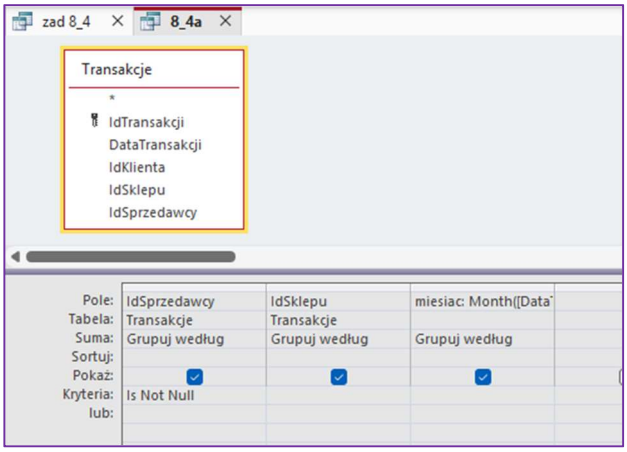

**Rysunek 18.**

Przejrzystym sposobem realizacji było utworzenie najpierw zbioru różnych trójek: sprzedawca – miesiąc – sklep, z pominięciem kas samoobsługowych (puste IdSprzedawcy). Dopiero na tak przygotowanych danych należało grupować po sprzedawcy i miesiącu oraz zliczać sklepy. Przykładowe rozwiązanie w MySQL przedstawiono na Rysunku 19.

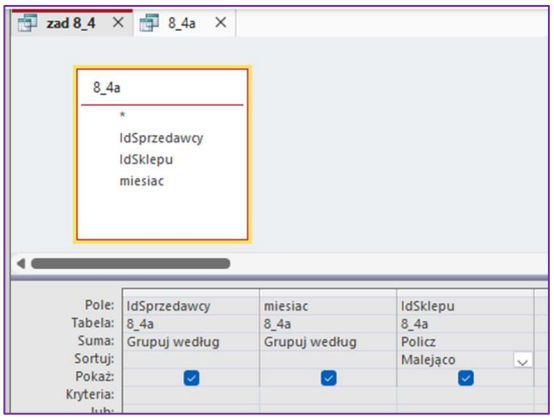
Podobną strategię można przyjąć rozwiązując zadanie w Access (Rysunki 20.-22.)



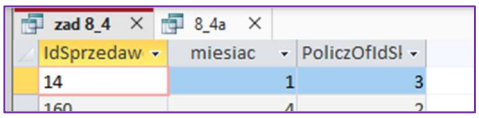

**Rysunek 19. Rozwiązanie w MySQL**




**Rysunek 20.**



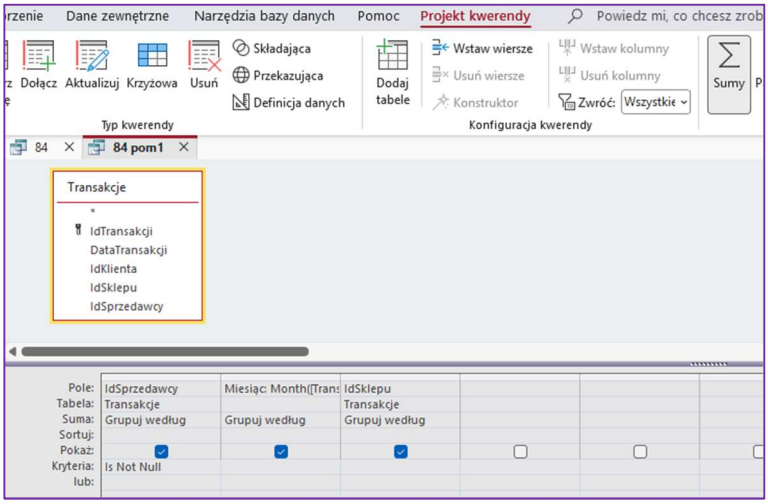

**Rysunek 21. Analogiczne rozwiązanie w Access**



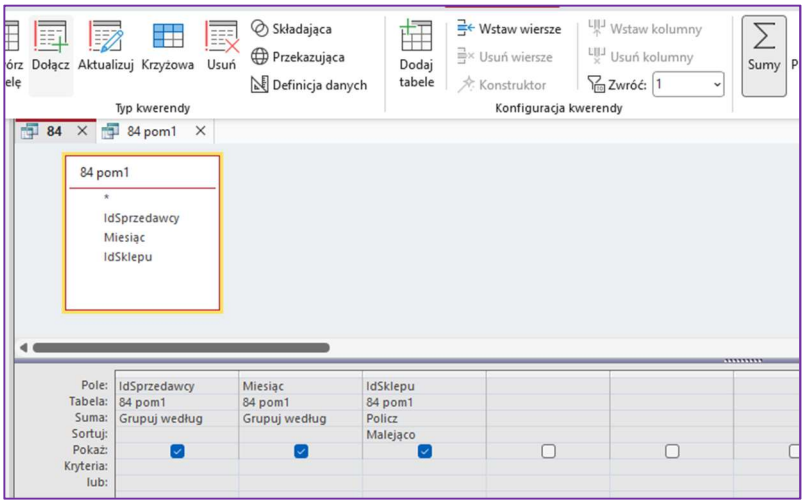

**Rysunek 22.**

## Zadanie 8.4. – przykładowe rozwiązania zdających

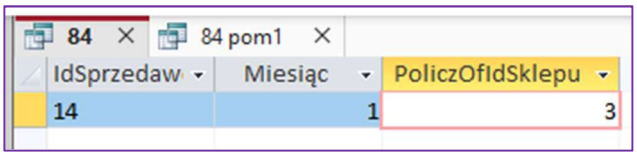
### Rozwiązanie 1.




**Rysunek 23.**



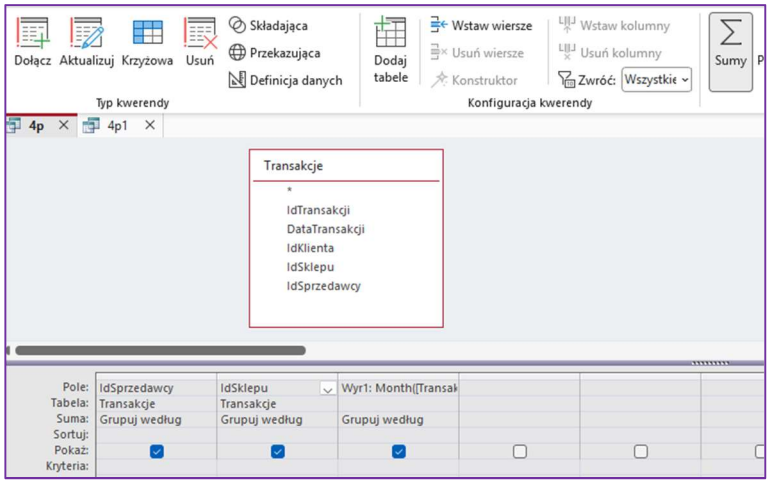

**Rysunek 24.**



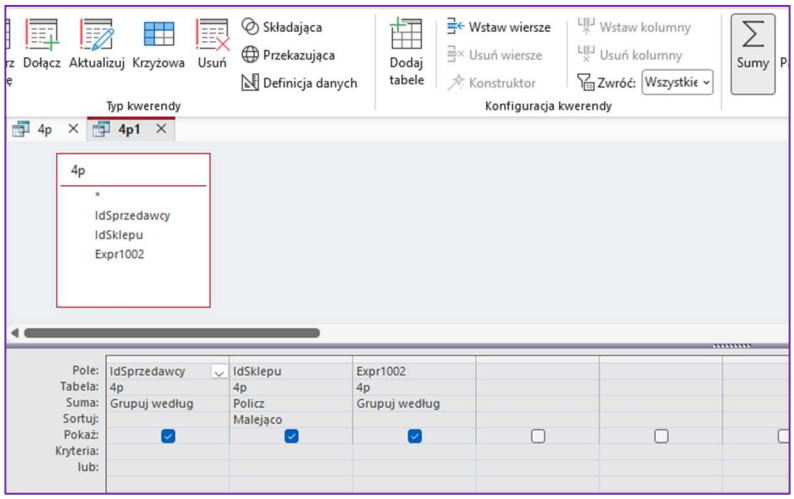

**Rysunek 25.**

Rozwiązanie z użyciem MS Access podobne do przedstawionego wcześniej, jedyna różnica to ograniczenie wyników do jednego wiersza (Rysunek 24. i Rysunek 25.) zawierającego odpowiedź.

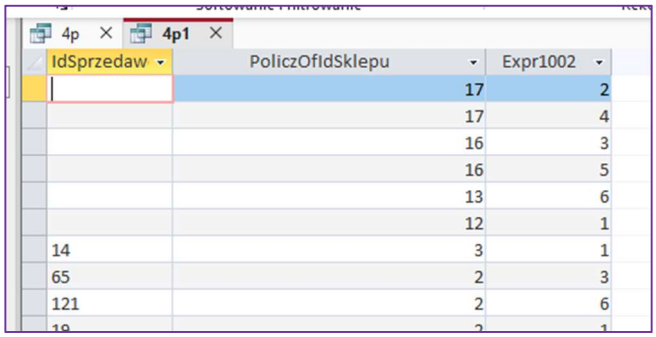
## Rozwiązanie 2.




**Rysunek 26.**




**Rysunek 27.**




**Rysunek 28.**

W powyższym rozwiązaniu zdający nie umieścił warunku, że idSprzedawcy musi być różne od pustego, ale w ostatecznej kwerendzie wynik pojawia się w pierwszym wierszu zawierającym dane sprzedawcy.
Część zdających pominęło to zadanie. Niektórym zdającym trudność sprawiło pobieranie miesiąca z daty.

Zadanie 8.5. (poziom wykonania 29%, Rysunek 29.) sprawdzało zapis zapytania SQL łączącego tabele *Kategorie*, *Produkty* i dane o zakupionych produktach. Warunek „zakupione produkty" oznaczał konieczność powiązania tabeli *Produkty* z tabelą *Opis_transakcji* – nie wszyscy zdający to zauważyli. Należało także zwrócić uwagę na to, że fragment „do ekspresu kolbowego" mógł znaleźć się na początku albo końcu opisu i użyć dwa razy znaku % (w MySQL) lub „*" (w przypadku zapisu z Access). Natomiast użycie DISTINCT, zapobiegającego wielokrotnemu wypisaniu tego samego produktu, nie było konieczne – w poleceniu nie znalazło się takie wymaganie.

**Zadanie 8.5. (0–2)**

Do bazy danych utworzonej na podstawie opisanych wcześniej plików dodano dwie kolejne tabele: o nazwie *Produkty* i o nazwie *Kategorie*, w których zapisano informacje o sprzedawanych produktach.

Tabela *Kategorie* zawiera następujące pola:

`IdKategorii` – unikatowy identyfikator kategorii produktu
`NazwaKategorii` – nazwa kategorii produktu.

Tabela *Produkty* zawiera następujące pola:

`IdProduktu` – unikatowy identyfikator produktu
`IdKategorii` – identyfikator kategorii, do której należy dany produkt
`Nazwa` – nazwa produktu
`Opis` – opis produktu.

Napisz w języku SQL zapytanie, w wyniku którego zostaną wypisane wszystkie zakupione produkty (IdProduktu i Nazwa) z kategorii (NazwaKategorii) „*spozywcze*" zawierające w opisie fragment: „*do ekspresu kolbowego*".

**Rysunek 29.**

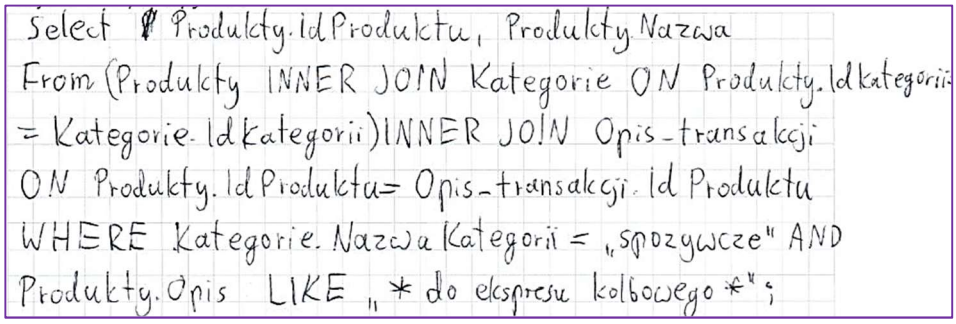
Przykładowe rozwiązanie:

```
SELECT DISTINCT Produkty.IdProduktu, Produkty.Nazwa
FROM (Kategorie INNER JOIN Produkty
      ON Kategorie.IdKategorii = Produkty.IdKategorii)
     INNER JOIN opis_transakcji
      ON Produkty.IdProduktu = opis_transakcji.IdProduktu
WHERE Kategorie.NazwaKategorii = 'spozywcze'
  AND Produkty.Opis LIKE '*do ekspresu kolbowego*';
```

W zależności od używanego systemu bazodanowego znakiem wieloznacznym w warunku LIKE może być * (np. w typowej składni programu Access) albo %. Istota rozwiązania pozostaje taka sama: połączyć właściwe tabele, ograniczyć rekordy do kategorii „spozywcze" i wyszukać wskazany fragment opisu.

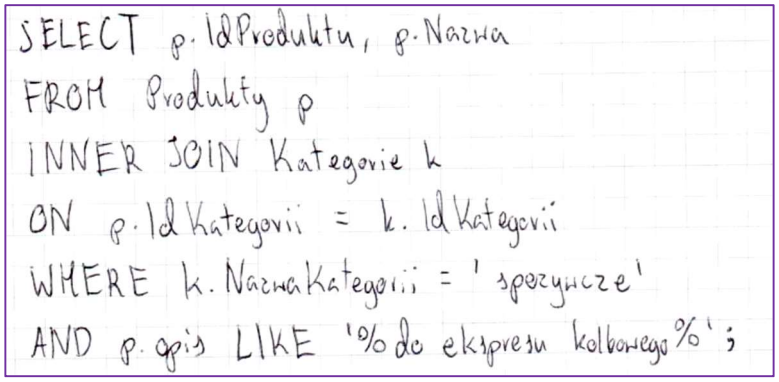
## Zadanie 8.5. – przykładowe rozwiązania zdających

**Rozwiązanie 1.** Poprawne zapytanie




**Rysunek 30.**

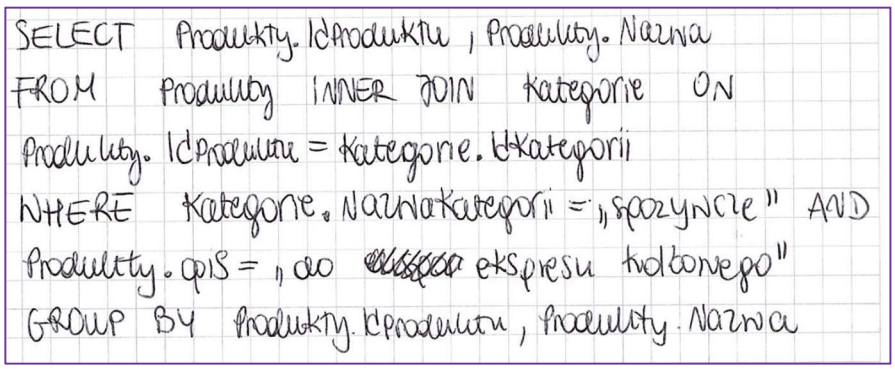
**Rozwiązanie 2.**




**Rysunek 31.**

W Rozwiązaniu 2. brakuje tabeli *Opis_transakcji*. To właśnie ona informuje, które produkty zostały faktycznie zakupione. Bez niej zapytanie może zwrócić również spełniający warunki produkt, który znajduje się w tabeli Produkty, ale nie wystąpił w żadnej transakcji. Za takie rozwiązanie można było uzyskać 1 pkt. Był to najczęstszy błąd popełniany w tym zadaniu przez zdających.

**Rozwiązanie 3.**




**Rysunek 32.**

W Rozwiązaniu 3. zdający popełnił kilka błędów. Przy łączeniu tabel *Produkty* i *Kategorie* zdający zapisał *Produkty.IdProduktu* = *Kategorie.IdKategorii* zamiast *Produkty.IdKategorii* = *Kategorie.IdKategorii*. Brakuje tabeli *Opis_transakcji*. Bez niej zapytanie wyszukuje produkty istniejące w tabeli *Produkty*, ale nie sprawdza, czy zostały one rzeczywiście zakupione.

Dodatkowo w *where* warunek dla *Opis* zapisano jako *Produkty.Opis = "do ekspresu kolbowego"* zamiast *Produkty.Opis LIKE "%do ekspresu kolbowego%"* albo *Produkty.Opis LIKE "*do ekspresu kolbowego*"*. Za takie rozwiązanie zdający otrzyma 0 punktów.

### Zadanie 6. – adresy IP

Zadanie 6. (Rysunek 33.) miało poziom wykonania 22% i wymagało podania długości adresów IP w wersjach 4 i 6. Było to zadanie krótkie, niezależne od pozostałych wiązek, sprawdzające podstawową wiedzę o sieciach komputerowych.

**Zadanie 6. (0–1)**

Uzupełnij zdania. Wpisz właściwe liczby bitów.

Adres IP w wersji 4 ma długość ..... bity.

Adres IP w wersji 6 ma długość ..... bitów.

**Rysunek 33.**

Prawidłowe uzupełnienie to: IPv4 – 32 bity, IPv6 – 128 bitów.

Niski poziom wykonania tego zadania na tle jego niewielkiej złożoności wskazuje, że w przygotowaniu do egzaminu warto systematycznie powtarzać również krótkie wymagania teoretyczne, a nie ograniczać pracy wyłącznie do programowania i zadań praktycznych.

### Zadania 7.3. i 7.4. – symulacja wzrostu rzęsy wodnej

Zadania 7.3. i 7.4. (poziom wykonania odpowiednio 32% i 31%, Rysunek 34.) wymagały poprawnego odwzorowania kolejności zdarzeń w symulacji.

**Informacja do zadań 7.3.–7.4.**

Na potrzeby zadania przyjmujemy, że w kolejnym roku przez 184 dni, od 1 marca 2023 do 31 sierpnia 2023, temperatury i opady utrzymywały się na stałym poziomie, co pozwalało na regularny wzrost rzęsy wodnej w tempie rozrostu 1,75% dziennie. Przyrost rzęsy następował w nocy, a pomiar zarośnięcia stawu – rano.

**1 marca 2023 rano** staw był zarośnięty rzęsą w 20%, tj. rzęsa wodna zajmowała 2000 $m^2$. W związku z tym, że staw nie powinien być zarośnięty w całości, właściciel postanowił pozbywać się jej nadmiaru. Do zbiornika wpuścił 80 amurów białych, z których każdy zjadał w ciągu dnia 0,25 $m^2$ rzęsy wodnej. Dodatkowo co piątek w ciągu dnia odławiał 60 $m^2$ rzęsy wodnej.

**Uwaga:** 30 kwietnia rano staw był zarośnięty w 25,79%.

**Zadanie 7.3. (0–2)**

Podaj, w którym dniu (licząc od 1 marca 2023) pomiar wykazał, że rzęsa wodna po raz pierwszy zajęła więcej niż 75% procent powierzchni stawu.

**Zadanie 7.4. (0–2)**

Podaj, jaka jest najmniejsza liczba amurów białych, jaką musi wpuścić właściciel, by rzęsa wodna w całym badanym okresie zajmowała maksymalnie 50% powierzchni stawu.

**Rysunek 34.**

Pomiar wykonywano rano, w ciągu dnia rzęsa była zjadana przez amury oraz – w piątki – dodatkowo odławiana, a przyrost o 1,75% następował w nocy. Podana w treści kontrolna wartość zarośnięcia 30 kwietnia (25,79%) umożliwiała sprawdzenie, czy kolejność operacji została zaimplementowana prawidłowo. Dla 80 amurów pierwszy poranny pomiar przekraczający 75% powierzchni stawu wypada w 167. dniu licząc od 1 marca, czyli 14 sierpnia 2023 r.

W zadaniu 7.4. należało dobrać minimalną liczbę amurów tak, aby żaden poranny pomiar w analizowanym okresie nie przekroczył 50%. Sprawdzenie kolejnych wartości prowadzi do minimum równego 94 amurom (dla 93 amurów warunek nie jest jeszcze spełniony).

## Zadania, z którymi zdający poradzili sobie najlepiej

Do tej grupy przyjęto zadania, których poziom wykonania był wyższy niż 60%. Takich zadań było 5:
• Zadanie 2.1. (92%, analiza dodawania pisemnego – otwarte)
• Zadanie 4.1. (82%, analiza hierarchii – otwarte)
• Zadanie 7.1. (78%, arkusz kalkulacyjny i wykres – praktyczne)
• Zadanie 1.1. (70%, analiza rekurencji – otwarte)
• Zadanie 8.1. (67%, baza danych – praktyczne).

## Zadanie 2.1. – dodawanie pisemne

Najłatwiejszym zadaniem w arkuszu było zadanie 2.1. (Rysunek 35.) Zdający mieli prześledzić dodawanie cyfr od prawej strony, uwzględniając przeniesienie z poprzedniej kolumny. Zadanie stanowiło bezpośrednie przygotowanie do zapisu algorytmu w zadaniu 2.2.

**Zadanie 2. Dodawanie**

Rozważamy dodawanie pisemne dwóch liczb zapisanych w systemie dziesiętnym, zilustrowane na przykładzie.

| | | | | | | |
|---|---|---|---|---|---|---|
| *Przeniesienie:* | | ***1*** | ***1*** | ***1*** | | ***1*** |
| Liczba *a*: | | | 2 | 7 | 7 | 3 | 2 |
| Liczba *b*: | + | | 7 | 2 | 6 | 1 | 9 |
| | | 1 | 0 | 0 | 3 | 5 | 1 |

W tym przykładzie mamy 4 przeniesienia.

**Zadanie 2.1. (0–1)**

Dla danych dwóch liczb: *a* i *b*, podaj, ile razy pojawiają się przeniesienia podczas ich dodawania.

| Liczba *a* | Liczba *b* | Liczba przeniesień |
|---|---|---|
| 37932 | 12528 | 3 |
| 88765 | 11111 | |
| 456789 | 222222 | |

**Rysunek 35.**

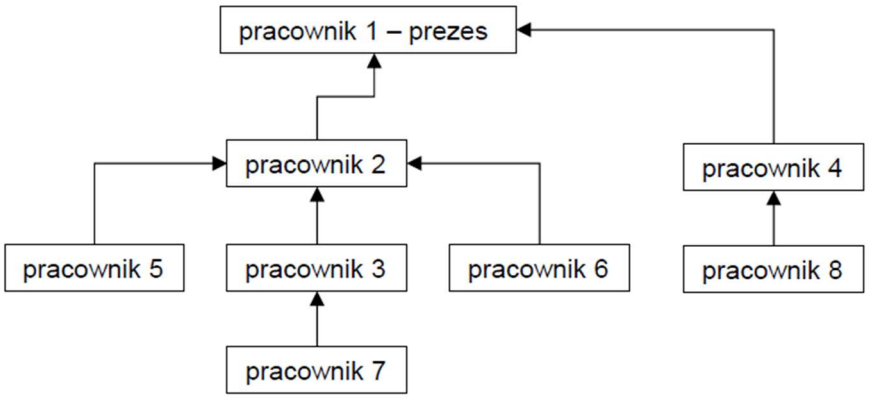
Dla podanych w tabeli par liczb liczby przeniesień wynoszą odpowiednio: 3, 0 i 3.
W zadaniu 2.2. (39%) należało tę samą procedurę uogólnić na dowolne dwie liczby o tej samej liczbie cyfr, i zapisać algorytm bez używania napisów ani tablic. Wystarczało wielokrotnie pobierać ostatnie cyfry za pomocą reszty z dzielenia przez 10, aktualizować przeniesienie i usuwać ostatnią cyfrę dzieleniem całkowitym przez 10.
Przykładowe rozwiązanie zadania 2.2.:

```
p = 0
przeniesienie = 0
dopóki a > 0 lub b > 0:
    s = (a mod 10) + (b mod 10) + przeniesienie
    jeżeli s >= 10:
        p = p + 1
        przeniesienie = 1
    w przeciwnym razie:
        przeniesienie = 0
    a = a div 10
    b = b div 10
```

## Zadanie 4.1. – odczytanie hierarchii

W zadaniu 4.1. (poziom wykonania 82%) należało odczytać z rysunku liczbę bezpośrednich oraz wszystkich podwładnych każdego pracownika. Było to zadanie wprowadzające do kolejnych zadań praktycznych z tej wiązki.

**Przykład 1.**




*Rysunek 1.* Przykład hierarchii w korporacji (strzałki wskazują na bezpośrednich przełożonych).
Przełożonymi pracownika 7 są pracownicy 3, 2 i 1. Bezpośrednim przełożonym pracownika 7 jest pracownik 3. Pracownik 3 (podobnie jak 5 oraz 6) jest bezpośrednim podwładnym pracownika 2, a podwładnymi pracownika 2 są pracownicy 3, 5, 6 i 7.

**Rysunek 36.**

**Zadanie 4.1. (0–1)**

Dla każdego pracownika z przykładowej hierarchii w korporacji (Rysunek 1.) określ, ilu ma on bezpośrednich podwładnych oraz ilu ma wszystkich podwładnych.

| Numer pracownika | Liczba bezpośrednich podwładnych | Liczba wszystkich podwładnych |
|---|---|---|
| 1 | | |
| 2 | 3 | 4 |
| 3 | | |
| 4 | | |
| 5 | | |
| 6 | | |
| 7 | | |
| 8 | 0 | 0 |

**Rysunek 37.**

Dla przedstawionej hierarchii pracownik 1 ma 2 bezpośrednich i 7 wszystkich podwładnych, pracownik 2 – odpowiednio 3 i 4, pracownik 3 – 1 i 1, pracownik 4 – 1 i 1, a pracownicy 5–8 nie mają podwładnych.

### Zadanie 7.1. – średnie temperatury i wykres

Zadanie 7.1. (poziom wykonania 78%) wymagało utworzenia zestawienia średnich miesięcznych temperatur i przedstawienia ich na wykresie kolumnowym. Było to typowe zadanie arkuszowe: oprócz poprawnego obliczenia średnich należało zaokrąglić wyniki do jednego miejsca po przecinku oraz zadbać o tytuł wykresu, opisy osi i czytelne oznaczenie miesięcy.

Poziom wykonania wskazuje, że zdający dobrze radzili sobie z podstawowymi narzędziami agregowania danych w arkuszu kalkulacyjnym oraz z przygotowaniem wykresu na podstawie gotowego zestawienia.

### Zadanie 1.1. – analiza rekurencji

W zadaniu 1.1. należało prześledzić kolejne wywołania funkcji rekurencyjnej. Zmiana argumentów była regularna: dla parzystego $n$ drugi argument był dzielony przez 2, a pierwszy mnożony przez 2; dla nieparzystego $n > 1$ drugi argument przyjmował wartość $(n-1)/2$.

**Zadanie 1. Rekurencja**

Dana jest zdefiniowana rekurencyjnie funkcja $A(m, n)$, gdzie $m$ i $n$ są dodatnimi liczbami całkowitymi.

$$A(m, n) = \begin{cases} m & \text{gdy } n = 1 \\ A\left(2 \cdot m, \frac{n}{2}\right) & \text{gdy } n > 1 \text{ oraz } n \text{ jest podzielne przez 2} \\ 2 \cdot A\left(m, \frac{n-1}{2}\right) + m & \text{gdy } n > 1 \text{ oraz } n \text{ nie jest podzielne przez 2} \end{cases}$$

**Zadanie 1.1. (0–3)**

Obliczenie wartości funkcji $A(3, 9)$ wprost z definicji wymaga trzech wywołań rekurencyjnych: $A(3, 4)$, $A(6, 2)$, $A(12, 1)$, ponieważ:

$$A(3, 9) = 2 \cdot A(3, 4) + 3 = 2 \cdot A(6, 2) + 3 = 2 \cdot A(12, 1) + 3 = 2 \cdot 12 + 3 = 27$$

Uzupełnij poniższą tabelę. Podaj liczbę wywołań rekurencyjnych funkcji $A$ oraz wypisz wywołania rekurencyjne wraz z ich argumentami (w ostatnim wierszu podaj tylko liczbę wywołań rekurencyjnych).

| ***m*** | ***n*** | **liczba wywołań rekurencyjnych funkcji *A*** | **wywołania rekurencyjne funkcji *A*** |
|---|---|---|---|
| 3 | 9 | 3 | $A(3, 4)$, $A(6, 2)$, $A(12, 1)$ |
| $2^5$ | $2^5$ | | |
| 10 | 15 | | |
| 1 | $2^{100} + 1$ | | |

**Rysunek 38.**

Rozwiązanie zadania i analiza definicji prowadzi do zależności $A(m,n) = m \cdot n$. Pozwala ona szybko rozwiązać zadanie 1.2.; mimo to jego poziom wykonania wyniósł tylko 37%, co wskazuje, że przejście od śledzenia pojedynczych wywołań do rozpoznania ogólnej własności funkcji było dużo trudniejsze.

Warto też zwrócić uwagę na to, że duża część zdających rozwiązuje tego typu zadania przepisując kod w środowisku programowania. Łatwiej wtedy otrzymać wyniki do zadania sformułowanego tak jak zadanie 1.1., ale niekoniecznie jest to najlepsze podejście w kontekście kolejnych zadań z wiązki. Samodzielne przeanalizowanie działania funkcji ułatwia przejście do uogólnienia i wyciągania wniosków.

## Zadanie 8.1. – agregowanie transakcji

Zadanie 8.1. (67%) było najłatwiejszym zadaniem z wiązki bazodanowej. Należało policzyć transakcje dla każdego klienta, wybrać klienta z największą liczbą transakcji, a następnie połączyć wynik z tabelą klientów, aby odczytać imię i nazwisko. W porównaniu z kolejnymi zadaniami nie było tu potrzeby obsługi braków danych, liczenia różnych wartości ani łączenia informacji z tabeli opisującej produkty.

## Wnioski i rekomendacje

1. W 2026 r. średni wynik wszystkich absolwentów wyniósł 40%. Absolwenci liceów ogólnokształcących uzyskali średnio 46%, a absolwenci techników 32%. Różnica między tymi grupami wyniosła 14 punktów procentowych.
2. Struktura trudności arkusza była zróżnicowana: 78% wszystkich punktów przypadało na zadania trudne lub bardzo trudne. Jednocześnie 8 punktów (16% puli) można było uzyskać w zadaniach łatwych lub bardzo łatwych.
3. Największe trudności sprawiały zdającym zadania łączące kilka etapów rozumowania lub przetwarzania danych. W przygotowaniu do egzaminu należy ćwiczyć rozkładanie problemów na etapy i dobór struktury danych do każdego etapu.
4. W zadaniach symulacyjnych trzeba szczególnie precyzyjnie ustalać kolejność zdarzeń. W zadaniu „Staw” poprawność implementacji można było kontrolować na wartości pośredniej podanej w treści (25,79% zarośnięcia 30 kwietnia). Podczas nauki warto wyrabiać nawyk weryfikowania rozwiązania na danych kontrolnych przed wykonaniem obliczeń dla całego zakresu.
5. Wynik zadania 6. pokazuje potrzebę systematycznego utrwalania podstawowej wiedzy teoretycznej. Krótkie informacje, takie jak długość adresu IPv4 i IPv6, mogą decydować o wynikach.
6. W zadaniach praktycznych należy konsekwentnie przypominać, że odpowiedź w pliku wynikowym musi wynikać z komputerowej realizacji rozwiązania, jeżeli treść zadania wymaga oddania pliku z programem, arkuszem lub bazą danych. W szczególności zaleca się:
   - zapisywać kody źródłowe pod nazwami jednoznacznie wskazującymi numer zadania
   - w arkuszu kalkulacyjnym umieszczać rozwiązania w czytelnie nazwanych i uporządkowanych arkuszach
   - w bazie danych zapisywać kwerendy w sposób umożliwiający łatwe powiązanie ich z numerem zadania
   - do rozwiązań z użyciem MySQL należy, zgodnie z procedurami, **koniecznie dołączać wyeksportowaną całą bazę w formacie sql** oprócz treści zapytań
   - przed zakończeniem egzaminu sprawdzać kompletność plików oddawanych do oceny.
7. Podczas przygotowania do egzaminu warto planować pracę nad zadaniami w taki sposób, aby obok ćwiczenia programowania rozwijać także analizę algorytmów, pracę z arkuszem kalkulacyjnym, relacyjnymi bazami danych i zagadnieniami teoretycznymi. Arkusz z 2026 r. pokazuje, że ostateczny wynik zależał od sprawnego łączenia tych obszarów.