INFORMATYKA

Poziom rozszerzony

Egzamin maturalny – Formuła 2023 · arkusz próbny nr 3

Imię i nazwisko: .....

WYBRANE:

.....
(system operacyjny)

.....
(program użytkowy)

.....
(środowisko programistyczne)

CZAS TRWANIA:	210 minut
LICZBA PUNKTÓW DO UZYSKANIA:	50
ZAŁĄCZNIK:	katalog DANE (archiwum matura-informatyka-pr-arkusz-3-dane.zip)

Arkusz przygotowany według zasad egzaminu maturalnego z informatyki w Formule 2023 (budowa arkusza, rodzaje zadań i sposób oddawania rozwiązań jak w arkuszach CKE 2023–2026; oprogramowanie zgodnie z komunikatem dyrektora CKE z 20 sierpnia 2026 r.). Zadania sprawdzają wymagania podstawy programowej w brzmieniu z Dz.U. 2024 poz. 1019. To nie jest arkusz CKE; treści, dane i historie są oryginalne. Materiał nie jest związany z CKE ani z producentami oprogramowania – nazwy programów, systemów i instytucji podano wyłącznie informacyjnie.

Instrukcja dla zdającego

1. Zanim zaczniesz, sprawdź, czy arkusz ma 19 stron (zadania 1–8) i czy masz katalog **DANE** z plikami do zadań (archiwum **matura-informatyka-pr-arkusz-3-dane.zip**). Jeśli czegoś brakuje, powiedz o tym nauczycielowi.
2. Na pierwszej stronie wpisz imię i nazwisko oraz wybrane przez siebie (na egzaminie – zadeklarowane) oprogramowanie: system operacyjny, program użytkowy i środowisko programistyczne. Według komunikatu dyrektora CKE z 20 sierpnia 2026 r. w 2027 r. do wyboru są: system **Windows** z pakietem Microsoft Office 2007 lub nowszym (Word, Excel, Access) albo z LibreOffice 7.0 lub nowszym (Writer, Calc, Base) lub OpenOffice/Apache OpenOffice 4.1 lub nowszym, albo system **Linux** z LibreOffice 7.0 lub nowszym lub OpenOffice/Apache OpenOffice 4.1 lub nowszym – zawsze razem z pakietem XAMPP (Apache, MySQL/MariaDB, PHP, phpMyAdmin) – oraz jeden język programowania: C/C++, Java (Java SE 8 lub nowsza) albo Python (3.10 lub nowszy).
3. Zadanie z symbolem  w nagłówku rozwiązujesz bez komputera – odpowiedź wpisz wyłącznie w arkuszu, w miejscu na nią przeznaczonym.
4. Programy pisz w wybranym języku programowania. Wszystkie utworzone przez siebie pliki źródłowe zapisz w katalogu nazwanym Twoim imieniem i nazwiskiem (na egzaminie katalog oznacza się numerem PESEL).
5. Jeżeli algorytm zapisywany w arkuszu przedstawiasz w języku programowania, a nie w pseudokodzie, użyj języka zadeklarowanego na egzamin (C++, Java lub Python).
6. Przy zadaniu z bazą danych w MySQL lub MariaDB zapisz w katalogu treści zapytań SQL, a przed końcem pracy wyeksportuj całą bazę do pliku *.sql. W programie Access lub Base oddajesz plik bazy z kwerendami nazwanymi tak, jak podano w zadaniu.
7. Każdy plik do oceny nazwij dokładnie tak, jak podano w zadaniu (rozszerzenie zależy od używanego programu) – pliki nazwane inaczej nie są sprawdzane. Zadanie praktyczne bez komputerowego rozwiązania otrzymuje 0 punktów.
8. Przed końcem czasu pracy zapisz w katalogu ostateczne wersje plików z rozwiązaniami.
9. Pisz czytelnie, długopisem lub piórem z czarnym tuszem (atramentem). Nie używaj korektora – błędny zapis wyraźnie przekreśl. Zapisów w brudnopisie nikt nie ocenia.
10. Kratki z punktacją przy zadaniach wypełnia egzaminator – nic w nich nie wpisuj.
11. Na egzaminie wtyczki AI generujące kod lub treść (np. Copilot) są w środowiskach programistycznych i w pakiecie biurowym odinstalowane albo wyłączone – rozwiązuj arkusz bez nich.
12. Programy do zadań 3. i 4. możesz dodatkowo sprawdzić na stronie **pierwszalawka.pl/kod-na-maturze** – wpisz kod zadania podany przy poleceniu. Narzędzie sprawdza wynik danego zadania (z numerem zadania albo bez), wypisany na ekran albo zapisany w pliku wyniki*.txt, na danych z arkusza i na dwóch innych zestawach danych.

PIERWSZA ŁAWKA

Zadanie 2. Reszta z dzielenia dużej liczby

Liczba dziesiętna N ma n cyfr: $c[1]$, $c[2]$, …, $c[n]$, gdzie $c[1]$ jest cyfrą najbardziej znaczącą. Liczba ta może mieć nawet tysiąc cyfr, dlatego nie da się jej zapisać w jednej zmiennej całkowitej. Chcemy obliczyć resztę z dzielenia N przez liczbę m.

Przydatna jest zależność: jeżeli r jest resztą z dzielenia liczby złożonej z pierwszych $i - 1$ cyfr przez m, to reszta z dzielenia liczby złożonej z pierwszych i cyfr przez m jest równa $(10 \cdot r + c[i]) \bmod m$ – tak jak w schemacie Hornera.

Zadanie 2.1. (0–1)

2.1.
0–1

Uzupełnij tabelę.

cyfry liczby N	m	$N \bmod m$
7 3 9 1 5	7	2
2 0 2 7 0 9 2 7	97	
9 9 9 9 9 9 9 9 9 9 9 9	1000	

Miejsce na obliczenia (brudnopis)

Zadanie 2.2. (0–4)

2.2.
0–1–2–3–4

Zapisz w pseudokodzie lub w wybranym języku programowania algorytm, który dla danych *n*, *c*[1..*n*] i *m* obliczy resztę z dzielenia liczby *N* przez *m*.

Uwaga: Twój algorytm może używać wyłącznie zmiennych przechowujących liczby całkowite i tablicy *c*; wartość żadnej zmiennej nie może w trakcie działania algorytmu przekroczyć 10 000. W zapisie algorytmu możesz korzystać tylko z operacji arytmetycznych: dodawania, odejmowania, mnożenia, dzielenia całkowitego (div) i reszty z dzielenia (mod), z porównań, instrukcji sterujących i przypisania. Zabronione jest używanie funkcji wbudowanych, potęgowania i zamiany typów (np. liczby na napis).

Specyfikacja:

Dane:

n – dodatnia liczba całkowita, $n \leq 1000$

c[1..n] – cyfry liczby *N* (liczby całkowite od 0 do 9), $c[1] \neq 0$

m – liczba całkowita, $2 \leq m \leq 1000$

Wynik:

r – reszta z dzielenia liczby *N* przez *m*

Miejsce na zapis algorytmu

Zadanie 3. Wyrażenia w ONP

W pliku **onp.txt** zapisano 500 wyrażeń w odwrotnej notacji polskiej, każde w osobnym wierszu. Tokeny – liczby całkowite od 1 do 20 oraz operatory **+**, **-** (odejmowanie) i ***** (mnożenie) – są oddzielone pojedynczymi spacjami. Wiersze numerujemy od 1.

Wyrażenie arytmetyczne w odwrotnej notacji polskiej (ONP) zapisujemy tak, że operator stoi po swoich dwóch argumentach, np. wyrażenie (3 + 4) · 2 ma w ONP postać **3 4 + 2 ***. Wartość wyrażenia w ONP oblicza się za pomocą stosu: tokeny czytamy od lewej do prawej; liczbę kładziemy na stos, a po napotkaniu operatora zdejmujemy ze stosu dwie liczby – najpierw *b*, potem *a* – i kładziemy na stos wynik działania *a* op *b* (np. dla **9 4 -** wynik to 5). Wyrażenie jest *poprawne*, jeżeli przy każdym operatorze na stosie są co najmniej dwie liczby, a po przeczytaniu wszystkich tokenów na stosie zostaje dokładnie jedna liczba – wartość wyrażenia. Na przykład wyrażenia **3 +** oraz **1 2** nie są poprawne.

token	3	4	+	2	*
stos po tokenie	3	3 4	7	7 2	14

Pierwsze wiersze pliku onp.txt:

```
6 10 8 + * 4 9 - * 12 - +
19 15 20 12 * - +
18 11 2 + 8 + -
12 1 - 13 - 10 7 + - 11 -
1
```

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.

Do Twojej dyspozycji jest plik **onp_przyklad.txt** w identycznym formacie. Odpowiedzi dla danych z tego pliku podano pod treściami zadań – możesz na nich sprawdzać działanie swojego programu.

Uwaga: Pamiętaj, że Twój program musi ostatecznie działać dla pliku **onp.txt**.

Zadanie 3.1. (0–3)

3.1.
0–1–2–3

Podaj liczbę poprawnych wyrażeń w pliku **onp.txt**.

Postać odpowiedzi: Jedna liczba całkowita.

Dla pliku **onp_przyklad.txt** poprawna odpowiedź to:

```
11
```

[kod zadania: inf-3-3a]

Zadanie 3.2. (0–3)

3.2.
0–1–2–3

Dla każdego poprawnego wyrażenia wyznacz największą liczbę elementów, które jednocześnie znajdowały się na stosie podczas obliczania jego wartości (np. dla wyrażenia 3 4 + 2 * jest to 2, a dla 1 2 3 * + jest to 3). Podaj największą z wyznaczonych liczb oraz liczbę poprawnych wyrażeń, dla których ją osiągnięto.

Postać odpowiedzi: Dwie liczby w jednym wierszu.

Dla pliku **onp_przyklad.txt** poprawna odpowiedź to:

```
5 1
```

[kod zadania: inf-3-3b]

Zadanie 3.3. (0–4)

3.3.
0–1–2–3–4

Oblicz wartości wszystkich poprawnych wyrażeń. Podaj największą z nich oraz numer wiersza, w którym zapisano wyrażenie o tej wartości. Jest jedno takie wyrażenie. Wartości wyrażeń i wszystkie wyniki pośrednie mieszczą się w zakresie liczb całkowitych 64-bitowych.

Postać odpowiedzi: Dwie liczby w jednym wierszu: największa wartość i numer wiersza.

Przykład: wartość wyrażenia **2 3 4 * -** wynosi $2 - 3 \cdot 4 = -10$.

Dla pliku **onp_przyklad.txt** poprawna odpowiedź to:

```
95 10
```

[kod zadania: inf-3-3c]

Do oceny oddajesz:

- plik tekstowy **wyniki3.txt** zawierający odpowiedzi do zadań 3.1.–3.3. (odpowiedź do każdego zadania powinna być poprzedzona jego numerem)
- pliki zawierające kody źródłowe Twoich programów o nazwach: **zad3_1** (zadanie 3.1.), **zad3_2** (zadanie 3.2.), **zad3_3** (zadanie 3.3.) – z rozszerzeniem zgodnym z językiem programowania (.py, .cpp lub .java); jeżeli jeden program rozwiązuje kilka zadań, zapisz go pod nazwą zawierającą numery tych zadań, np. **zad3_1_2** (uwaga: brak tych plików jest równoznaczny z brakiem rozwiązania zadania).

Zadanie 4. Kolejka w urzędzie

Urząd otwiera się o godzinie 8:00. W pliku **klienci.txt** zapisano dane 600 klientów w kolejności ich przychodzenia. W każdym wierszu są dwie liczby całkowite oddzielone spacją: moment przyjścia klienta (liczba sekund, które upłynęły od otwarcia urzędu) oraz czas jego obsługi w sekundach. Momenty przyjścia tworzą ciąg niemalejący. Klientów numerujemy od 1 (numer klienta = numer wiersza). Klienci są obsługiwani w kolejności przyjścia. Jeżeli okienko jest wolne, obsługa klienta zaczyna się w chwili jego przyjścia; w przeciwnym razie klient czeka, aż okienko się zwolni (obsługa może zacząć się dokładnie w chwili zakończenia obsługi poprzedniego klienta).

Przykład: klienci przychodzą w chwilach 0, 10 i 100, a ich obsługa trwa odpowiednio 50, 30 i 20 sekund. Przy jednym okienku klient 1 jest obsługiwany od 0 do 50, klient 2 – od 50 do 80 (czekał 40 sekund), a klient 3 – od 100 do 120 (nie czekał).

Pierwsze wiersze pliku klienci.txt:

```
127 94
168 64
289 69
348 47
362 20
```

Napisz program (lub kilka programów), który(-e) znajdzie(-ą) i da(dzą) odpowiedzi do podanych zadań. Odpowiedzi do poszczególnych zadań zapisz w pliku **wyniki4.txt**. Każdą odpowiedź poprzedź numerem oznaczającym zadanie.

Do Twojej dyspozycji jest plik **klienci_przyklad.txt** w identycznym formacie. Odpowiedzi dla danych z tego pliku podano pod treściami zadań – możesz na nich sprawdzać działanie swojego programu.

Uwaga: Pamiętaj, że Twój program musi ostatecznie działać dla pliku **klienci.txt**.

Zadanie 4.1. (0–2)

4.1.
0–1–2

Urząd ma jedno okienko. Podaj moment (liczbę sekund od otwarcia urzędu), w którym zakończy się obsługa ostatniego klienta.

Postać odpowiedzi: Jedna liczba całkowita.

Dla pliku **klienci_przyklad.txt** poprawna odpowiedź to:

```
513
```

[kod zadania: inf-3-4a]

Zadanie 4.2. (0–3)

4.2.
0–1–2–3

Urząd ma jedno okienko. Czas oczekiwania klienta to różnica między momentem rozpoczęcia jego obsługi a momentem jego przyjścia. Podaj najdłuższy czas oczekiwania oraz numer klienta, który czekał najdłużej (jeżeli jest kilku takich klientów – podaj najmniejszy numer).

Postać odpowiedzi: Dwie liczby w jednym wierszu: czas oczekiwania i numer klienta.

Dla pliku **klienci_przyklad.txt** poprawna odpowiedź to:

```
226 8
```

[kod zadania: inf-3-4b]

Zadanie 4.3. (0–4)

4.3.
0–1–2–3–4

Urząd otworzył drugie okienko. Każdy kolejny klient (w kolejności przyjścia) jest kierowany do tego okienka, w którym jego obsługa może zacząć się najwcześniej; jeżeli w obu okienkach może zacząć się w tej samej chwili, klient idzie do okienka nr 1. Podaj, ilu klientów zostało obsłużonych w okienku nr 1, a ilu – w okienku nr 2.

Postać odpowiedzi: Dwie liczby w jednym wierszu: liczba klientów okienka nr 1 i okienka nr 2.

Przykład: dla trzech klientów z przykładu podanego wyżej klient 1 trafia do okienka nr 1 (obsługa od 0 do 50), klient 2 – do okienka nr 2 (obsługa od 10 do 40), a klient 3 – do okienka nr 1 (w obu okienkach jego obsługa może zacząć się w chwili 100). Odpowiedź: 2 1.

Dla pliku **klienci_przyklad.txt** poprawna odpowiedź to:

```
3 5
```

[kod zadania: inf-3-4c]

Do oceny oddajesz:

- plik tekstowy **wyniki4.txt** zawierający odpowiedzi do zadań 4.1.–4.3. (odpowiedź do każdego zadania powinna być poprzedzona jego numerem)
- pliki zawierające kody źródłowe Twoich programów o nazwach: **zad4_1** (zadanie 4.1.), **zad4_2** (zadanie 4.2.), **zad4_3** (zadanie 4.3.) – z rozszerzeniem zgodnym z językiem programowania (.py, .cpp lub .java); jeżeli jeden program rozwiązuje kilka zadań, zapisz go pod nazwą zawierającą numery tych zadań, np. **zad4_1_2** (uwaga: brak tych plików jest równoznaczny z brakiem rozwiązania zadania).

PIERWSZA ŁAWKA

Zadanie 7. Kawiarnia

W pliku **kawiarnia.txt** zapisano 3804 pozycji sprzedaży małej kawiarni z okresu od 1 marca do 31 maja 2026 r. Pierwszy wiersz pliku jest wierszem nagłówkowym, a dane w wierszach rozdzielono znakami tabulacji. W każdym wierszu zapisano:

Data – data sprzedaży (rrrr-mm-dd)
Godzina – godzina sprzedaży (gg:mm)
Produkt – nazwa produktu
Kategoria – kategoria produktu: kawa, napoje albo wypieki
Liczba – liczba sprzedanych sztuk
Cena – cena jednej sztuki w zł (z dwoma miejscami po przecinku)

Przykład:

Data	Godzina	Produkt	Kategoria	Liczba	Cena
2026-03-01	07:12	americano	kawa	3	11,00
2026-03-01	07:24	cappuccino	kawa	1	14,00
2026-03-01	07:30	flat white	kawa	2	15,00
2026-03-01	07:44	cappuccino	kawa	2	14,00

Utargiem nazywamy łączną wartość sprzedaży (liczba sztuk · cena). Z wykorzystaniem danych zawartych w pliku oraz dostępnych narzędzi informatycznych wykonaj podane zadania. Odpowiedzi zapisz w pliku **wyniki7.txt**. Odpowiedź do każdego zadania poprzedź numerem tego zadania.

Zadanie 7.1. (0–2)

7.1.
0–1–2

Podaj utarg w każdej kategorii produktów oraz łączny utarg kawiarni w całym okresie.

Zadanie 7.2. (0–3)

7.2.
0–1–2–3

Wykonaj zestawienie utargu w poszczególnych dniach tygodnia (poniedziałek, wtorek, …, niedziela – łącznie w całym okresie). Na podstawie zestawienia utwórz wykres kolumnowy. Pamiętaj o czytelnym opisie wykresu: tytuł, opisy osi, nazwy dni tygodnia na osi X.

Zadanie 7.3. (0–2)

7.3.
0–1–2

Podaj pełną godzinę (np. 8 oznacza okres od 8:00 do 8:59), w której w całym okresie sprzedano łącznie najwięcej sztuk produktów, oraz tę liczbę sztuk.

Zadanie 7.4. (0–3)

7.4.
0–1–2–3

Do przygotowania jednej porcji napoju kawowego zużywa się kawy ziarnistej: espresso – 8 g, americano – 8 g, cappuccino – 16 g, latte – 16 g, flat white – 16 g. Rano 1 marca 2026 r. kawiarnia miała 6000 g kawy. Każdego dnia po zamknięciu kawiarni właściciel sprawdza zapas kawy; jeżeli jest mniejszy niż 3000 g, zamawia 5000 g kawy, którą dostarcza się następnego dnia przed otwarciem. Podaj, ile razy kawiarnia zamówiła kawę w okresie od 1 marca do 31 maja 2026 r. (wlicz także ewentualne zamówienie złożone 31 maja), oraz datę złożenia ostatniego zamówienia.

Przykład: jeżeli rano zapas wynosi 3400 g, a w ciągu dnia sprzedano 20 porcji cappuccino, 10 porcji espresso i 5 porcji latte, to zużyto 20 · 16 + 10 · 8 + 5 · 16 = 480 g kawy. Po zamknięciu zapas

wynosi 2920 g, czyli mniej niż 3000 g – właściciel zamawia kawę i następnego dnia rano zapas wynosi 7920 g.

Do oceny oddajesz:

- plik tekstowy **wyniki7.txt** zawierający odpowiedzi do zadań 7.1.–7.4. (odpowiedź do każdego zadania poprzedzona jego numerem)
- plik zawierający wykres do zadania 7.2. o nazwie **wykres7_2** (z rozszerzeniem .xlsx albo .ods)
- plik z komputerową realizacją rozwiązań zadań 7.1.–7.4. o nazwie **zad7** (.xlsx albo .ods); jeżeli któreś z zadań rozwiązujesz programem – także plik z kodem źródłowym o nazwie **zad7_N** (N – numer zadania, z rozszerzeniem zgodnym z językiem programowania) (uwaga: brak tych plików jest równoznaczny z brakiem rozwiązania zadania)

CO JESZCZE JEST W ŚRODKU

Matura 2027 z informatyki – poziom rozszerzony. Arkusz próbny 3 z kluczem

Podgląd pokazał 10 z 32 stron. Pełna wersja: 32 stron A4, PDF do druku i DOCX do edycji.

- Arkusz próbny: Matura 2027 z informatyki – poziom rozszerzony, zestaw nr 3.
- Poziom rozszerzony · Formuła 2023 · arkusz próbny na maturę 2027 · szybkie potęgowanie, schemat Hornera, wyrażenia w ONP, kolejka w urzędzie, systemy liczbowe, kryptografia, kawiar
- Czas pracy: 210 minut · 8 zadań · 50 pkt · 32 strony A4.
- W pakiecie plików: arkusz w PDF (do druku), ten sam arkusz w DOCX (do edycji w Wordzie) oraz archiwum DANE z plikami .txt do zadań przy komputerze (z plikami *_przyklad.txt do spra
- Na końcu: zasady oceniania, klucz odpowiedzi i przykładowe realizacje zadań otwartych.
- Arkusz jest zbudowany jak matura CKE 2023–2026: 8 zadań, około 20% punktów bez komputera, odpowiedzi zapisywane w plikach wynikiN.txt, wiązka arkusza kalkulacyjnego z wykresem i wi
- Programy z zadań 3 i 4 możesz sprawdzić automatycznie na pierwszalawka.pl/kod-na-maturze – przy każdym poleceniu jest kod zadania.
- „Kod na maturze" uruchamia Twój program w Pythonie, C++ albo Javie na danych z arkusza i na dwóch dodatkowych zestawach.
- To nie jest arkusz CKE ani kopia arkusza z poprzednich lat – wszystkie teksty i zadania są oryginalne, przygotowane zgodnie z wymaganiami egzaminacyjnymi na rok 2027.
- Dzięki temu uczeń nie zna zadań z internetu, a wynik rzetelnie pokazuje stan przygotowania.

Kup pełną wersję – plik od razu na e-mail

pierwszalawka.pl/material/matura-informatyka-pr-2027-arkusz-probny-3