CENTRALNA KOMISJA EGZAMINACYJNA

Arkusz zawiera informacje prawnie chronione do momentu rozpoczęcia egzaminu.

WYPEŁNIA ZDAJĄCY

KOD

PESEL

Miejsce na naklejkę.
Sprawdź, czy kod na naklejce to **E-100**.
Jeżeli tak – przyklej naklejkę.
Jeżeli nie – zgłoś to nauczycielowi.

EGZAMIN MATURALNY Z INFORMATYKI

POZIOM ROZSZERZONY

CZĘŚĆ I

DATA: **14 czerwca 2022 r.**
GODZINA ROZPOCZĘCIA: **9:00**
CZAS PRACY: **60 minut**
LICZBA PUNKTÓW DO UZYSKANIA: **15**

WYPEŁNIA ZDAJĄCY

WYBRANE:

.....
(system operacyjny)

.....
(program użytkowy)

.....
(środowisko programistyczne)

Instrukcja dla zdającego

1. Sprawdź, czy arkusz egzaminacyjny zawiera 10 stron (zadania 1–3). Ewentualny brak zgłoś przewodniczącemu zespołu nadzorującego egzamin.
2. Rozwiązania i odpowiedzi zapisz w miejscu na to przeznaczonym przy każdym zadaniu.
3. Pisz czytelnie. Używaj długopisu/pióra tylko z czarnym tuszem/atramentem.
4. Nie używaj korektora, a błędne zapisy wyraźnie przekreśl.
5. Pamiętaj, że zapisy w brudnopisie nie będą oceniane.
6. Wpisz zadeklarowane (wybrane) przez Ciebie na egzamin system operacyjny, program użytkowy oraz środowisko programistyczne.
7. Na tej stronie oraz na karcie odpowiedzi wpisz swój numer PESEL i przyklej naklejkę z kodem.
8. Nie wpisuj żadnych znaków w części przeznaczonej dla egzaminatora.

EINP-R1-**100**-2206

Układ graficzny
© CKE 2021

Zadanie 1. Liczby nudne i ciekawe

Rozważmy operację, która dodatniej liczbie całkowitej przyporządkowuje sumę kwadratów jej cyfr w zapisie dziesiętnym. Przykładowo: liczbie 123 zostanie przyporządkowana liczba 14, ponieważ $1^2 + 2^2 + 3^2 = 14$. Utwórzmy teraz ciąg, którego pierwszym elementem będzie dodatnia liczba całkowita *n,* a każdy kolejny jego element to wynik zastosowania powyższej operacji do elementu poprzedzającego go w tym ciągu.
Jeśli w otrzymanym w ten sposób ciągu pojawi się liczba 1, to początkową liczbę *n* nazywamy liczbą *nudną*, w przeciwnym razie *n* nazywamy liczbą *ciekawą*.

Przykład:
Dla *n* = 13 otrzymujemy ciąg:
13, $10 = 1^2 + 3^2$, $1 = 1^2 + 0^2$
Tak więc 13 jest liczbą *nudną*.

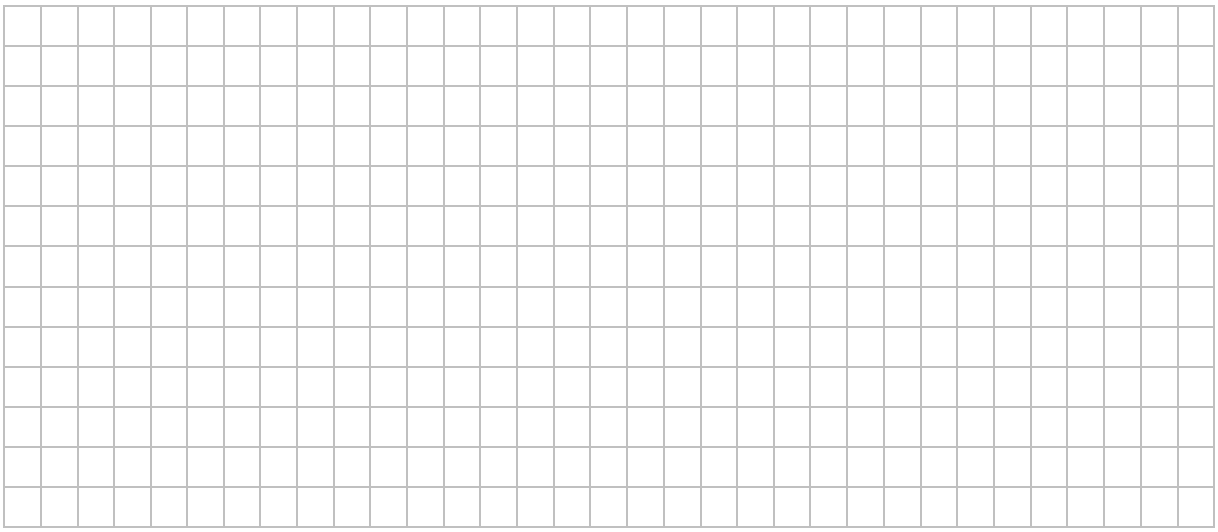
Liczba 4 jest liczbą *ciekawą*, ponieważ:
4, $16 = 4^2$, $37 = 1^2 + 6^2$, $58 = 3^2 + 7^2$, $89 = 5^2 + 8^2$, $145 = 8^2 + 9^2$, $42 = 1^2 + 4^2 + 5^2$,
$20 = 4^2 + 2^2$, **4** $= 2^2 + 0^2$, 16, 37, 58, 89, 145, 42, 20, **4**, … itd.
Czyli nigdy nie otrzymamy liczby 1.

Zadanie 1.1. (0–2)

Uzupełnij tabelę – wpisz TAK, jeśli podana liczba jest *nudna*, albo NIE – jeśli nie jest *nudna*.

n	**Czy *nudna*?**
4	Nie
229	
82	

Miejsce na obliczenia:



Zadanie 1.2. (0–2)

W postaci pseudokodu lub w wybranym języku programowania napisz funkcję **SumaKwCyfr**(*n*), która dla dodatniej liczby całkowitej *n* oblicza sumę kwadratów jej cyfr.

Przykład:
Dla *n* = 123 wynikiem **SumaKwCyfr**(123) jest liczba $1^2 + 2^2 + 3^2 = 14$.

Uwaga: W zapisie funkcji możesz korzystać tylko z instrukcji sterujących, operatorów arytmetycznych: dodawania, odejmowania, mnożenia, dzielenia, dzielenia całkowitego i reszty z dzielenia; operatorów logicznych, porównań i instrukcji przypisywania lub samodzielnie napisanych funkcji i procedur wykorzystujących powyższe operacje. **Zabronione** jest używanie funkcji wbudowanych dostępnych w językach programowania.

Specyfikacja:
Dane:

n – dodatnia liczba całkowita

Wynik:

dodatnia liczba całkowita – suma kwadratów cyfr liczby *n* w zapisie dziesiętnym

Miejsce na zapis funkcji:

Zadanie 1.3. (0–3)

W postaci pseudokodu lub w wybranym języku programowania napisz algorytm, który dla danej dodatniej liczby całkowitej $n < 1000$ sprawdza, czy liczba n jest *ciekawa* czy *nudna*.

Uwaga:

- w algorytmie możesz użyć funkcji **SumaKwCyfr**(*n*) z poprzedniego zadania
- możesz skorzystać z faktu, że suma kwadratów cyfr liczby trzycyfrowej jest nie większa niż $9^2 + 9^2 + 9^2 = 243$

Uwaga: W zapisie możesz korzystać tylko z instrukcji sterujących, operatorów arytmetycznych: dodawania, odejmowania, mnożenia, dzielenia, dzielenia całkowitego i reszty z dzielenia; operatorów logicznych, porównań, odwoływania się do pojedynczych elementów tablicy i instrukcji przypisywania lub samodzielnie napisanych funkcji i procedur wykorzystujących powyższe operacje. **Zabronione** jest używanie funkcji wbudowanych dostępnych w językach programowania.

Specyfikacja:

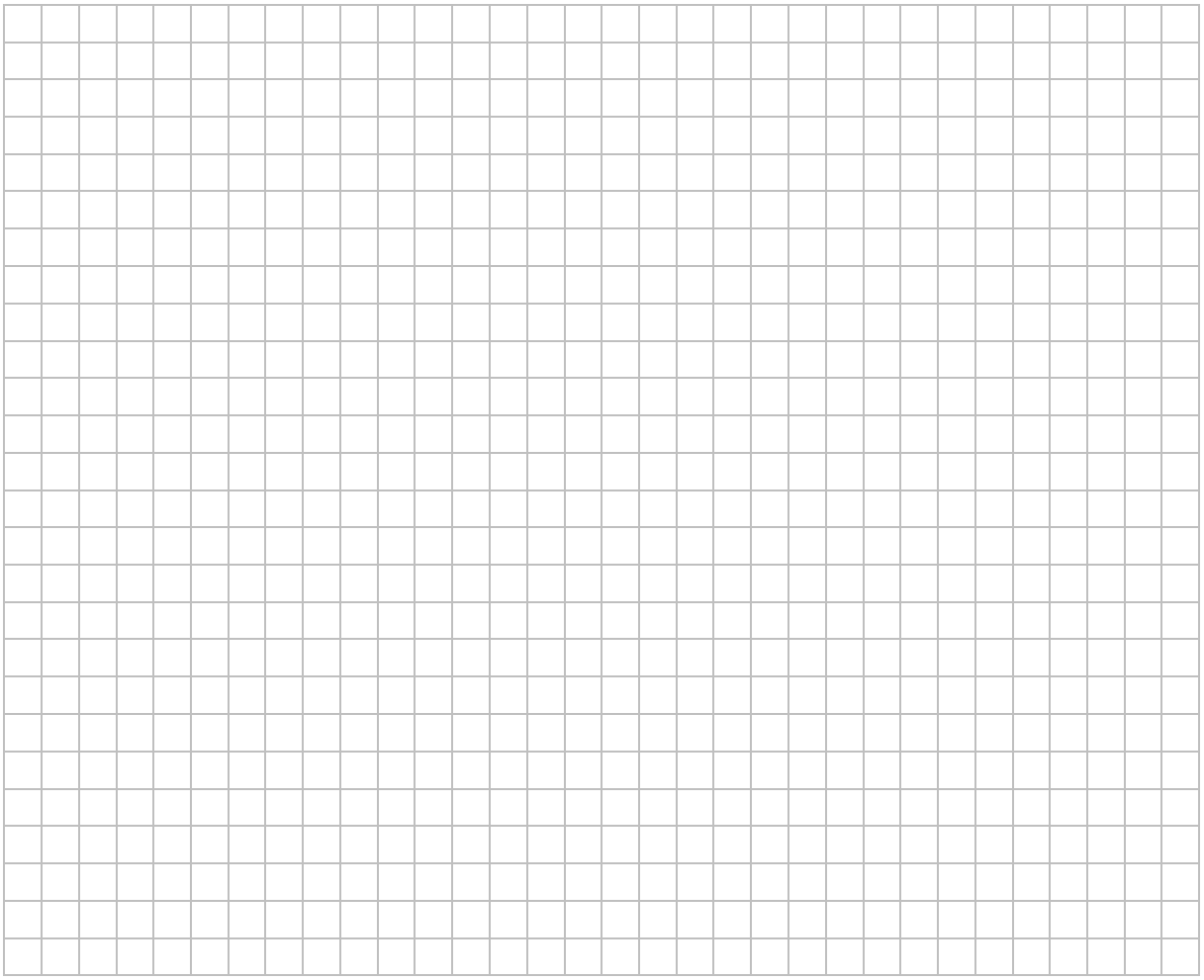
Dane:

n – dodatnia liczba całkowita mniejsza od 1 000

Wynik:

Prawda – gdy liczba jest *nudna*, albo Fałsz – gdy jest *ciekawa* (nie jest *nudna*)

Miejsce na zapis algorytmu:



Zadanie 2. Funkcja Koduj

Dana jest funkcja **Koduj**(*n*), która dla zadanej dodatniej liczby całkowitej *n* oblicza pewien jej kod – słowo puste lub słowo zbudowane tylko z wielkich liter A lub B.

Specyfikacja

Dane*:*

n – dodatnia liczba całkowita

Wynik*:*

Kod liczby *n* – słowo puste lub słowo zbudowane z wielkich liter A lub B

Funkcja Koduj(*n*):

jeżeli *n* = 1

wynikiem jest ''

w przeciwnym wypadku

k ← *n* div 2

jeżeli *k* mod 2 = 0

wynikiem jest Koduj(*k*) + 'A'

w przeciwnym wypadku

wynikiem jest 'B' + Koduj(*k*)

Uwaga:

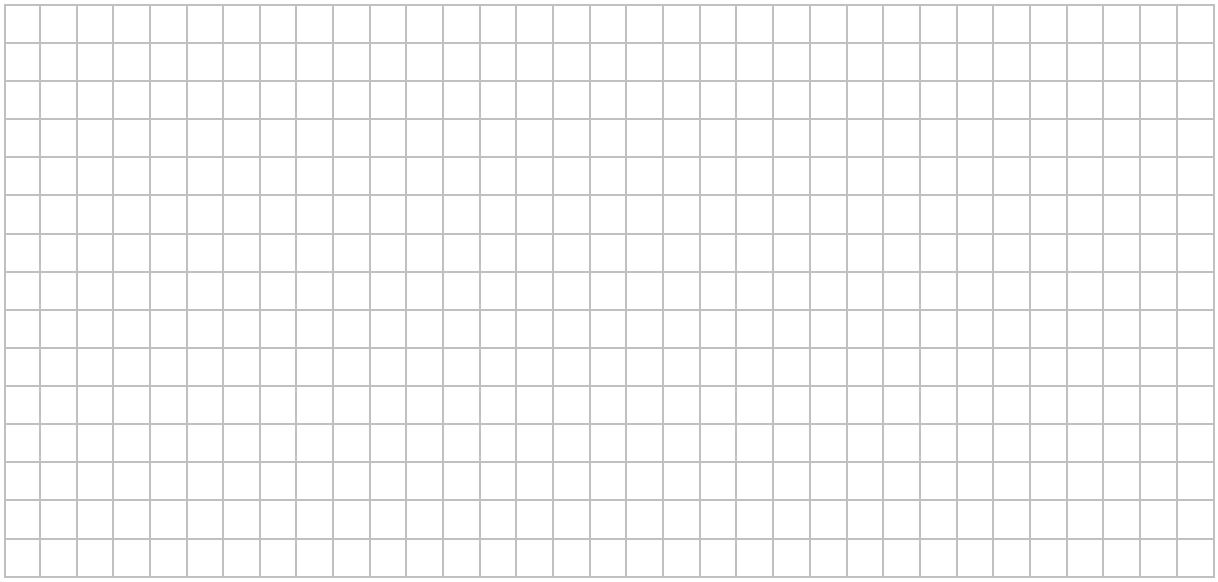
- *div* jest operatorem oznaczającym część całkowitą z dzielenia
- *mod* jest operatorem oznaczającym resztę z dzielenia
- *słowem* nazywamy dowolny ciąg znaków
- '' oznacza słowo puste (bez liter)
- + jest operatorem łączącym znak i słowo lub dwa słowa w jedno słowo.

Zadanie 2.1. (0–2)

Uzupełnij tabelę – wpisz wynik działania funkcji **Koduj**(*n*) dla podanych wartości *n.*

n	**Wynik działania funkcji Koduj(*n*)**
1	''
2	B
12	
33	
158	

Miejsce na obliczenia:

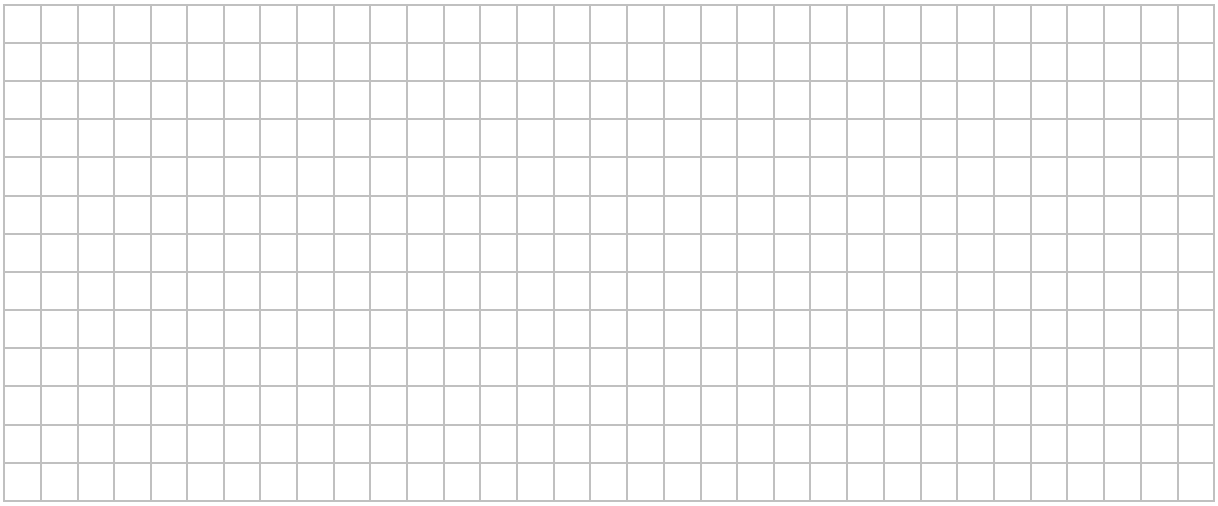


Zadanie 2.2. (0–2)

Uzupełnij tabelę – dla podanych wartości *n* wpisz liczbę wszystkich wywołań funkcji **Koduj** przy pierwszym wywołaniu **Koduj**(*n*).

n	**Pierwsze wywołanie funkcji Koduj**	**Liczba wszystkich wywołań funkcji Koduj**
1	Koduj(1)	1
2	Koduj(2)	2
12	Koduj(12)	
33	Koduj(33)	
1022	Koduj(1022)	

Miejsce na obliczenia:



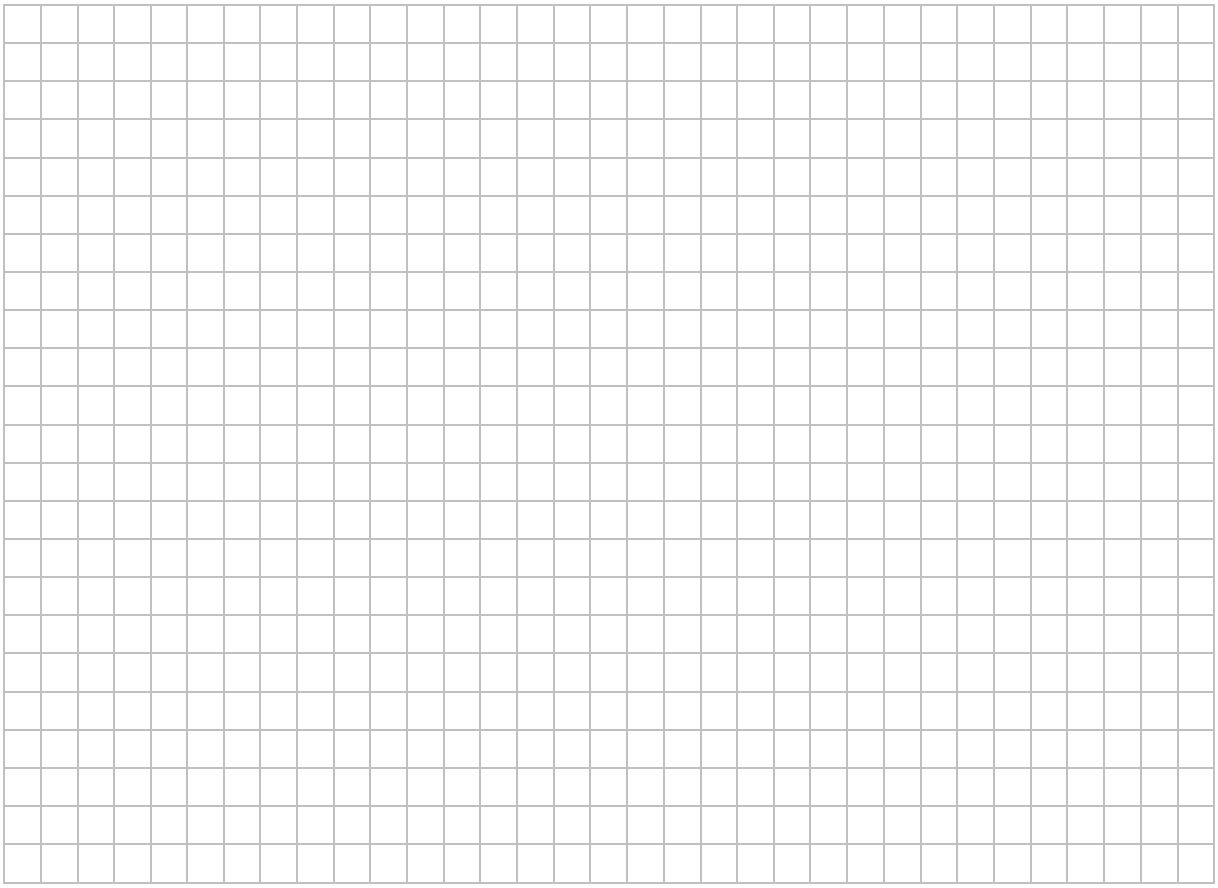
Zadanie 2.3. (0–2)

Podaj dwie różne dodatnie liczby całkowite, dla których funkcja **Koduj** da ten sam kod złożony z sześciu znaków.

Liczba 1: _____

Liczba 2: _____

Miejsce na obliczenia:



Zadanie 3. Test

Oceń prawdziwość podanych zdań. Zaznacz **P**, jeśli zdanie jest prawdziwe, albo **F** – jeśli jest fałszywe.
W każdym zadaniu punkt uzyskasz tylko za komplet poprawnych odpowiedzi.

Zadanie 3.1. (0–1)

Po dodaniu dwóch liczb 101101_2 i 111011_2 zapisanych w systemie binarnym otrzymamy:

1.	1101000_2	**P**	**F**
2.	68_{16}	**P**	**F**
3.	140_8	**P**	**F**
4.	1120_4	**P**	**F**

Zadanie 3.2. (0–1)

Poniżej przedstawiono opis dwóch tabel danych zawierających informacje o lekarzach i wizytach u tych lekarzy.

Pole *Id_lekarza* w tabeli *Lekarze* jest połączone relacją „jeden do wielu” z polem *Id_lekarza* w tabeli *Wizyty*.

Lekarze

Nazwa pola	Typ	Klucz
Id_lekarza	Tekst(5)	Klucz główny
Imie_lekarza	Tekst(50)	
Nazwisko_lekarza	Tekst(50)	
Specjalnosc_lekarza	Tekst(200)	

Wizyty

Nazwa pola	Typ	Klucz
Id_lekarza	Tekst(5)	Klucz obcy
Id_pacjenta	Tekst(10)	
Data_wizyty	Data	

1.	Wynikiem zapytania: **SELECT** Imie_lekarza, Nazwisko_lekarza, count(*) **FROM** Lekarze **JOIN** Wizyty **ON** Lekarze.Id_lekarza = Wizyty.Id_lekarza **GROUP BY** Wizyty.Id_lekarza; jest zestawienie zawierające imię i nazwisko każdego lekarza oraz liczbę wizyt u tego lekarza.	**P**	**F**
2.	Wynikiem zapytania: **SELECT** Specjalnosc_lekarza, count(*) **FROM** Lekarze **JOIN** Wizyty **ON** Lekarze.Id_lekarza = Wizyty.Id_lekarza **GROUP BY** Lekarze.Id_lekarza; jest zestawienie zawierające nazwy specjalności oraz łączne liczby wizyt u lekarzy tych specjalności.	**P**	**F**
3.	Wynikiem zapytania: **SELECT** Specjalnosc_lekarza, count(Specjalnosc_lekarza) **FROM** Lekarze **JOIN** Wizyty **ON** Lekarze.Id_lekarza = Wizyty.Id_lekarza **GROUP BY** Wizyty.Id_lekarza; jest zestawienie zawierające nazwy specjalności oraz łączne liczby wizyt u lekarzy tych specjalności.	**P**	**F**
4.	Wynikiem zapytania: **SELECT** Id_pacjenta, count(*) **FROM** Wizyty **JOIN** Lekarze **ON** Lekarze.Id_lekarza = Wizyty.Id_lekarza **GROUP BY** Lekarze.Id_lekarza; jest zestawienie zawierające liczby wizyt pacjentów u poszczególnych lekarzy.	**P**	**F**

BRUDNOPIS (*nie podlega ocenie*)