

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: **20 maja 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 9 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**-2205

Zadanie 1. n -permutacja

Dla dodatniej liczby całkowitej n , **n -permutacją** nazywamy taki n -elementowy ciąg liczb całkowitych, który zawiera każdą z liczb $1, 2, \dots, n$ dokładnie jeden raz.

Przykład:

ciąg $(4, 2, 1, 3)$ jest *4-permutacją*,

ciąg $(6, 5, 4, 1, 2, 3)$ jest *6-permutacją*,

ciągi $(1, 3, 1, 2)$ i $(2, 3, 4, 5)$ nie są *4-permutacjami*.

W ciągu n liczb całkowitych, który nie jest *n -permutacją*, można podmienić niektóre elementy tak, aby otrzymać *n -permutację*.

Przykład:

w ciągu $(1, 3, 1)$ wystarczy podmienić jeden element – pierwszą lub ostatnią jedynkę (1) – na dwójkę (2), aby powstały ciąg był *3-permutacją*.

Zadanie 1.1. (0–2)

Uzupełnij poniższą tabelę – dla każdego z podanych ciągów podaj najmniejszą liczbę elementów, które trzeba podmienić, aby dany ciąg był *n -permutacją*. Jeśli ciąg jest już *n -permutacją*, wpisz 0.

n	ciąg	liczba elementów do podmiany
3	$(1, 3, 1)$	1
4	$(1, 4, 2, 5)$	
5	$(2, 2, 2, 2, 2)$	
4	$(4, 2, 3, 1)$	
6	$(5, 4, 1, 5, 6, 8)$	
6	$(8, 4, 9, 6, 5, 7)$	

Zadanie 1.2. (0–4)

Zapisz w pseudojęzyku lub wybranym języku programowania algorytm, który dla danego ciągu n dodatnich liczb całkowitych zapisanego w tablicy A obliczy najmniejszą liczbę elementów, które trzeba w nim podmienić, aby otrzymać *n -permutację*.

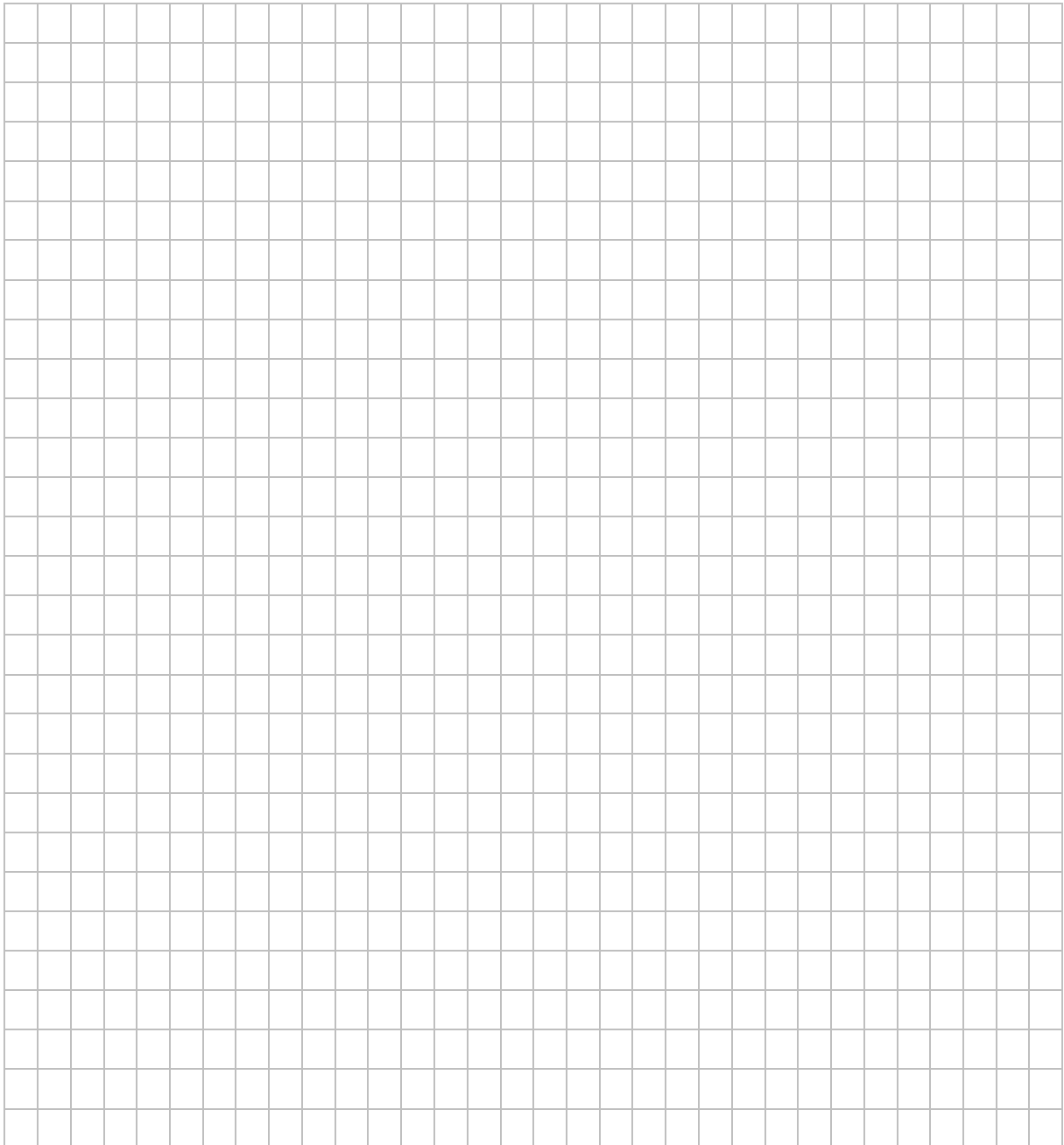
Uwaga: W zapisie algorytmu możesz korzystać tylko z instrukcji sterujących, operatorów arytmetycznych: dodawania, odejmowania, mnożenia, dzielenia, dzielenia całkowitego i reszty z dzielenia; operatorów logicznych, porównań, odwoływania się do pojedynczych elementów tablicy i instrukcji przypisania lub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

Specyfikacja:

Dane:

 n – dodatnia liczba całkowita $A[1..n]$ – tablica n dodatnich liczb całkowitych, gdzie $A[i]$ jest i -tym elementem ciągu

Wynik:

 k – minimalna liczba elementów, które trzeba podmienić w ciągu zapisanym w tablicy A , aby otrzymać n -permutację**Algorytm:**

Wypełnia egzaminator	Nr zadania	1.1.	1.2.
	Maks. liczba pkt.	2	4
	Uzyskana liczba pkt.		

Zadanie 2. ab-słowo

Niech n będzie dodatnią liczbą całkowitą i niech s będzie słowem o długości n zbudowanym z liter a lub b . Zapis $s[i]$ oznacza i -tą literę w tym słowie ($1 \leq i \leq n$).

Dla słowa s wykonujemy poniższy algorytm. Wynikiem działania algorytmu jest wartość zmiennej k .

Algorytm

$A[0] \leftarrow 0$

dla $i = 1, 2, \dots, n$

jeżeli $s[i] = 'a'$

$A[i] \leftarrow A[i-1] + 1$

w przeciwnym razie

$A[i] \leftarrow A[i-1]$

$B[n+1] \leftarrow 0$

dla $j = n, n-1, \dots, 1$

jeżeli $s[j] = 'b'$

$B[j] \leftarrow B[j+1] + 1$

w przeciwnym razie

$B[j] \leftarrow B[j+1]$

$k \leftarrow 1$

dla $i = 0, 1, 2, \dots, n$

jeżeli $A[i] + B[i+1] > k$

$k \leftarrow A[i] + B[i+1]$

Zadanie 2.1. (0–2)

Uzupełnij tabelę – wpisz wynik działania algorytmu dla podanych wartości s .

n	s	Wynik działania algorytmu (wartość k)
5	<i>aabab</i>	4
2	<i>ab</i>	2
3	<i>aaa</i>	3
6	<i>aababb</i>	
9	<i>baabbaaab</i>	

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)

Dany jest algorytm:

```

s ← 0
dla i = 1, 2, ..., n
  dla j = i, i + 1, ..., n
    s ← s + 1
  
```

Złożoność obliczeniowa powyższego algorytmu oceniona liczbą wykonań instrukcji $s \leftarrow s + 1$, w zależności od dodatniej liczby całkowitej n , jest

1.	liniowa.	P	F
2.	kwadratowa.	P	F
3.	$n \log n$.	P	F
4.	nie większa niż sześcienna.	P	F

Zadanie 3.2. (0–1)

Po dodaniu liczb 132_4 oraz 3111_4 zapisanych w systemie czwórkowym otrzymamy:

1.	1111011_2	P	F
2.	362_8	P	F
3.	$F3_{16}$	P	F
4.	3303_4	P	F

Wypełnia egzaminator	Nr zadania	2.3.	3.1.	3.2.
	Maks. liczba pkt.	2	1	1
	Uzyskana liczba pkt.			

Zadanie 3.3. (0–1)

W bazie danych istnieje tabela *mandaty*(*numer*, *id_osoby*, *punkty*) zawierająca następujące dane:

numer	id_osoby	punkty
1	1	5
2	1	14
3	2	20
4	3	21
5	2	1
6	1	2

1.	<p>Wynikiem zapytania:</p> <pre>SELECT id_osoby, sum(punkty) FROM mandaty GROUP BY id_osoby HAVING sum(punkty) > 5</pre> <p>jest zestawienie:</p> <pre>1 14 2 20 3 21</pre>	P	F
2.	<p>Wynikiem zapytania:</p> <pre>SELECT id_osoby, sum(punkty) FROM mandaty GROUP BY id_osoby</pre> <p>jest zestawienie:</p> <pre>1 21 2 21 3 21</pre>	P	F
3.	<p>Wynikiem zapytania:</p> <pre>SELECT numer + punkty FROM mandaty</pre> <p>jest</p> <pre>86</pre>	P	F
4.	<p>Wynikiem zapytania:</p> <pre>SELECT count(punkty) FROM mandaty WHERE punkty = 21</pre> <p>jest</p> <pre>1</pre>	P	F

Wypełnia egzaminator	Nr zadania	3.3.
	Maks. liczba pkt.	1
	Uzyskana liczba pkt.	

BRUDNOPIS (*nie podlega ocenie*)

