

Arkusz zawiera informacje prawnie chronione do momentu rozpoczęcia egzaminu.

, © CKE 2013	UZUPE	ŁNIA ZDAJĄCY	mieisce
graficzny	KOD	PESEL	miejsce na naklejkę
Układ gr			
Þ			

EGZAMIN MATURALNY Z INFORMATYKI

POZIOM ROZSZERZONY

CZĘŚĆ I

Instrukcja dla zdającego

- 1. Sprawdź, czy arkusz egzaminacyjny zawiera 9 stron. Ewentualny brak zgłoś przewodniczącemu zespołu nadzorującego egzamin.
- 2. Rozwiązania i odpowiedzi zamieść w miejscu na to przeznaczonym.
- 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 podlegają ocenie.
- 6. Wpisz obok zadeklarowane (wybrane) przez Ciebie na egzamin środowisko komputerowe, kompilator języka programowania oraz program użytkowy.
- 7. Jeżeli rozwiązaniem zadania lub jego części jest algorytm, to zapisz go w wybranej przez siebie notacji: listy kroków, schematu blokowego lub języka programowania, który wybierasz na egzamin.
- 8. Na tej stronie oraz na karcie odpowiedzi wpisz swój numer PESEL i przyklej naklejkę z kodem.
- 9. Nie wpisuj żadnych znaków w części przeznaczonej dla egzaminatora.



10 maja 2017

Godzina rozpoczęcia: 14:00

WYBRANE:
(środowisko)
(kompilator)
(program użytkowy)

Czas pracy: 90 minut

Liczba punktów do uzyskania: 20

MIN-R1_**1**P-172

Zadanie 1. Sortowanie

Rozważmy problem sortowania ciągu liczb całkowitych z przedziału [1..*k*] dla znanej całkowitej wartości *k*. Poniżej prezentujemy algorytm rozwiązujący ten problem, zgodny z następującą specyfikacją:

Specyfikacja:

Dane: n, k – liczby całkowite dodatnie

T[1..n] – ciąg liczb całkowitych z zakresu [1..k]

Wynik: W[1..n] – uporządkowany niemalejąco ciąg liczb z tablicy T[1..n]

Algorytm Sortowanie

```
dla i=1..k wykonuj
    Liczba_wystapien[i] \leftarrow 0
dla i=1..n wykonuj
    Liczba_wystapien[T[i]] \leftarrow Liczba_wystapien[T[i]] + 1

p \leftarrow 1
dla j=1..k wykonuj
    dla i=1..Liczba_wystapien[j] wykonuj
    W[p] \leftarrow j
    p \leftarrow p+1
```

Zadanie 1.1 (0-1)

Uzupełnij poniższą tabelę – podaj końcową zawartość tablicy *Liczba_wystapien* dla odpowiednich danych wejściowych.

n	k	T[1n]	Końcowa zawartość <i>Liczba_wystapien</i> [1k]
10	5	[1, 2, 3, 4, 5, 1, 2, 3, 4, 4]	[2, 2, 2, 3, 1]
5	10	[1, 3, 3, 5, 10]	
5	5	[5, 5, 5, 5, 5]	
10	4	[1, 2, 3, 1, 2, 3, 1, 1, 2, 3]	

Miejsce na obliczenia.



Zadanie 1.2 (0-2)

Rangą elementu T[i] w ciągu T[1..n] nazywać będziemy liczbę elementów ciągu T[1..n], które są mniejsze od T[i].

Przykład:

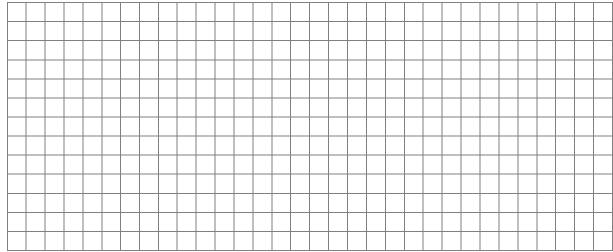
Dla n=10, k=5 oraz T[1..10] = [1, 2, 3, 4, 5, 1, 4, 3, 2, 5] mamy:

- ranga elementu T[8] (T[8] = 3) jest równa 4, gdyż w ciągu T[1..10] występują cztery elementy mniejsze od T[8]: dwa razy występuje liczba 1 i dwa razy występuje liczba 2;
- ranga T[10] (T[10] = 5) jest równa 8, gdyż w ciągu T[1..10] występuje osiem liczb mniejszych od T[8];
- ranga T[6] (T[6] = 1) jest równa 0.

Przyjmij, że tablica *Liczba_wystapien* ma zawartość uzyskaną po wykonaniu algorytmu *Sortowanie*. Na podstawie tego faktu uzupełnij poniższy algorytm w taki sposób, aby po jego wykonaniu wartość zmiennej r była równa randze elementu T[i], dla ustalonego i $(1 \le i \le n)$:

$$r \leftarrow 0$$
dla j=1.. wykonuj
$$r \leftarrow r+Liczba_wystapien[j]$$

Miejsce na obliczenia.



Zadanie 1.3 (0–3)

Rozważmy algorytm, w którym teraz elementy tablicy T mogą być dowolnymi dodatnimi liczbami całkowitymi.

Algorytm LicznikiMod

dla i=1..k wykonuj
 Liczba_wystapien[i]
$$\leftarrow$$
 0
dla i=1..n wykonuj
 m \leftarrow 1+(T[i] mod k)
 Liczba_wystapien[m] \leftarrow Liczba_wystapien[m] + 1
w \leftarrow Liczba wystapien[1]

Uzupełnij poniższą tabelę:

n	k	<i>T</i> [1n]	Końcowa zawartość <i>Liczba_wystapien</i> [1k]
10	2	[1, 2, 3, 4, 5, 1, 2, 3, 4, 4]	[5, 5]
10	3	[1, 2, 3, 4, 5, 1, 2, 3, 4, 4]	
10	4	[1, 2, 3, 4, 5, 1, 2, 3, 4, 4]	
10	5	[1, 2, 3, 4, 5, 1, 2, 3, 4, 4]	

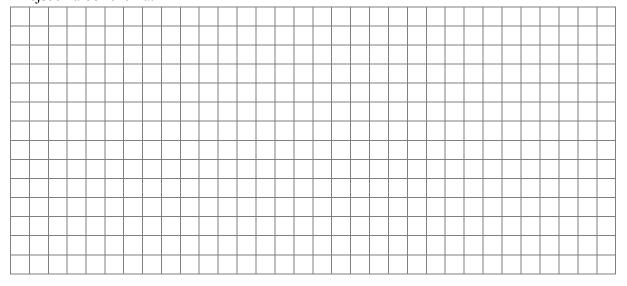
Uzupełnij specyfikację algorytmu *LicznikiMod*:

Dane: n, k – liczby całkowite dodatnie

T[1..n] – tablica liczb całkowitych dodatnich

Wynik: *w* –

Miejsce na obliczenia.



	Nr zadania	1.1.	1.2.	1.3.
Wypełnia	Maks. liczba pkt.	1	2	3
egzaminator	Uzyskana liczba pkt.			

Zadanie 2. Ciąg Pentanacciego

Rozważmy ciąg liczb p_0, p_1, p_2, \dots zdefiniowany w następujący sposób:

$$\begin{cases} p_0 = 0 \\ p_1 = 1 \\ p_2 = 1 \\ p_3 = 2 \\ p_4 = 4 \\ p_n = p_{n-1} + p_{n-2} + p_{n-3} + p_{n-4} + p_{n-5} \, dla \, n \geq 5 \end{cases}$$

Zadanie 2.1 (0-2)

Uzupełnij poniższą tabelę.

n	p_n
5	8
7	
9	

Zadanie 2.2 (0-3)

Poniżej prezentujemy algorytm, który powinien wyznaczać *n*-ty element podanego ciągu. Uzupełnij luki w algorytmie tak, aby jego działanie było zgodne z podaną specyfikacją.

Specyfikacja:

Dane: n – nieujemna liczba całkowita Wynik: w – liczba całkowita równa p_n

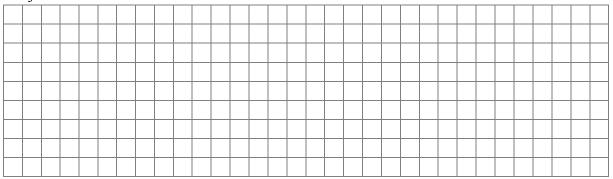
Algorytm:

 $tab[0] \leftarrow 0$ $tab[1] \leftarrow 1$ $tab[2] \leftarrow 1$ $tab[3] \leftarrow 2$ $tab[4] \leftarrow 4$ $i \leftarrow 5$ $dopóki i \leq \dots wykonuj$ $temp \leftarrow tab[0] + tab$

$$temp \leftarrow tab[0] + tab[1] + tab[2] + tab[3] + tab[4]$$
$$tab[.... \mod 5] \leftarrow temp$$
$$i \leftarrow i+1$$

Uwaga: a mod b oznacza resztę z dzielenia liczby a przez liczbę b.

Miejsce na obliczenia.



Zadanie 2.3 (0-3)

Rozważmy poniższy ciąg r_n :

$$\begin{cases} r_0 = 0 \\ r_1 = 1 \\ r_2 = 1 \\ r_3 = 0 \\ r_4 = 0 \end{cases}$$

$$r_4 = 0$$
 Find the primary state and the state of the primary state and the primary state and the primary state of the primary state and the prim

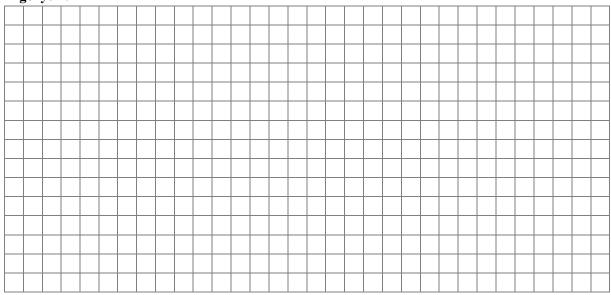
Zauważmy, że liczba p_n jest parzysta wtedy i tylko wtedy, gdy r_n =0. Można też sprawdzić, że wartości r_n powtarzają się cyklicznie – każda wartość jest taka sama jak wartość wcześniejsza o sześć wyrazów – a zatem wartość r_n zależy wyłącznie od liczby n mod 6. Na podstawie tego faktu podaj algorytm **o jak najmniejszej złożoności obliczeniowej**, który działa zgodnie z poniższą specyfikacją.

Specyfikacja:

Dane: n – nieujemna liczba całkowita

Wynik: w - 0 (zero), gdy liczba p_n jest parzysta, natomiast 1 (jeden), gdy liczba p_n jest nieparzysta

Algorytm:



	Nr zadania	2.1.	2.2.	2.3.
agzaminator	Maks. liczba pkt.	2	3	3
	Uzyskana liczba pkt.			

Zadanie 3. Test

W każdym z poniższych zadań oceń, które z podanych zdań są prawdziwe. Zaznacz P, jeśli zdanie jest prawdziwe, lub F – jeśli jest fałszywe.

Zadanie 3.1 (0-1)

Dane są tablica A[1..6] o zawartości [6, 2, -1, 5, 1, 2] oraz następujący fragment algorytmu:

 $s \leftarrow 0$

 $n \leftarrow 3$

 $i \leftarrow 6$

dopóki i > n - 1 wykonuj

 $s \leftarrow s + A[i]$

 $i \leftarrow i - 1$

Po wykonaniu tego algorytmu spełniony jest warunek

s jest parzyste.	P	F
s = 7.	P	F
s > 6.	P	F
s=3.	P	F

Zadanie 3.2 (0-1)

Realizacji usług poczty elektronicznej służy protokół

reminerable distance between the property provider		
SMTP.	P	F
IMAP.	P	F
EMAIL.	P	F
POP3.	P	F

Zadanie 3.3 (0-1)

Liczbą większą od 150(10) jest

10011001(2)	P	F
1222(4)	P	F
227(8)	P	F
9B ₍₁₆₎	P	F

Zadanie 3.4 (0-1)

Obrazy rastrowe

są reprezentowane jako tablice pikseli, co powoduje istotną utratę jakości przy powiększaniu obrazu.	P	F
tworzone są przy użyciu wyrażeń matematycznych opisujących występujące w obrazie odcinki, krzywe, elipsy itp.	P	F
mogą być wprowadzane do komputera przy użyciu urządzeń takich jak aparat cyfrowy lub skaner.	P	F
mogą powstać w efekcie cyfrowego zapisu obrazu widzialnego.	P	F

Zadanie 3.5 (0-1)

Algorytm zwany sitem Eratostenesa opierający się na "wykreślaniu" wielokrotności kolejnych (niewykreślonych wcześniej) liczb naturalnych służy wyznaczeniu

największego wspólnego dzielnika dwóch liczb.	P	F
najmniejszej wspólnej wielokrotności dwóch liczb.	P	F
liczb pierwszych z zadanego przedziału.	P	F
potęg dwójki z zadanego przedziału.	P	F

Zadanie 3.6 (0-1)

Przykładem programu, który służy do tłumaczenia instrukcji kodu źródłowego **programu komputerowego** na język maszynowy, jest

walidator.	P	F
kompilator.	P	F
edytor tekstu.	P	F
defragmentator.	P	F

Wypelnia egzaminator	Nr zadania	3.1.	3.2.	3.3.	3.4.	3.5.	3.6.
	Maks. liczba pkt.	1	1	1	1	1	1
	Uzyskana liczba pkt.						

BRUDNOPIS (nie podlega ocenie)