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

TEST DIAGNOSTYCZNY

TERMIN: **marzec 2021 r.**
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 8 stron (zadania 1–3). Ewentualny brak zgłoś przewodniczącemu zespołu nadzorującego egzamin.
2. 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**-2103

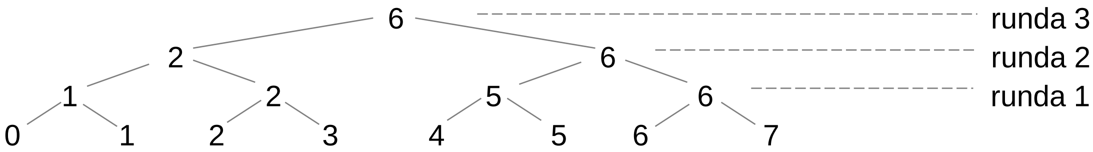
Układ graficzny
© CKE 2021

Zadanie 1. Turniej

W turnieju siatkówki bierze udział n drużyn ponumerowanych kolejnymi liczbami całkowitymi od 0 do $n - 1$, gdzie $n = 2^k$ dla pewnej liczby całkowitej $k > 0$. Turniej odbywa się w rundach systemem pucharowym – przegrywający odpada z turnieju. W każdej rundzie drużyny grają w parach i do dalszej rundy przechodzi tylko zwycięzca meczu. W każdej rundzie mecze są ponumerowane kolejnymi liczbami całkowitymi, poczynając od 1. W pierwszej rundzie w meczu nr 1 grają drużyny 0 i 1, w meczu nr 2 – drużyny 2 i 3, w meczu nr 3 – drużyny 4 i 5, w meczu nr i – drużyny $2*(i-1)$ oraz $2*(i-1)+1$, itd. W każdej z kolejnych rund w meczu nr 1 grają zwycięzcy meczów o numerach 1 i 2 z poprzedniej rundy, w meczu nr 2 – zwycięzcy meczów o numerach 3 i 4 z poprzedniej rundy, w meczu nr i – zwycięzcy meczów o numerach $2*i - 1$ oraz $2*i$ z poprzedniej rundy itd. Turniej trwa dokładnie k rund.

Przykład

Przykładową rozgrywkę w turnieju 8-drużynowym przedstawiono w postaci drzewa na rysunku poniżej. Na najniższym poziomie rysunku drzewa zapisano numery drużyn, natomiast w węzłach wewnętrznych – numery zwycięskich drużyn w poszczególnych meczach. Zwycięzcą turnieju została drużyna nr 6, która w meczu finałowym pokonała drużynę o numerze 2.



Numer rundy, w której mogą zmierzyć się dwie drużyny o numerach x i y, można wyznaczyć z zapisów binarnych liczb x i y o długości k (liczba rund). Twoim zadaniem jest odkrycie tej zależności.

Zadanie 1.1. (0–2)

Dla podanej liczby k (liczba rund w turnieju) oraz numerów drużyn x i y wyznacz numer rundy w turnieju, w której te dwie drużyny mogą się zmierzyć ze sobą.

k	*x*	*y*	*x* dwójkowo	*y* dwójkowo	nr rundy, w której mogą się zmierzyć drużyny x i *y*
3	2	6	010	110	3
4	0	3	0000	0011	2
4	3	7	0011	0111	
5	16	30	10000	11110	

Miejsce na obliczenia:

Zadanie 1.2. (0–4)

Napisz algorytm (w pseudokodzie lub w wybranym języku programowania), który dla danych liczb całkowitych *k*, *x* i *y* obliczy numer rundy w turnieju dla 2^k drużyn, w której mogą się spotkać drużyny *x* i *y*.

Uwaga: W zapisie algorytmu możesz korzystać wyłącznie z instrukcji sterujących, operatorów arytmetycznych (w tym dzielenia całkowitego i dzielenia z resztą), operatorów logicznych, porównań i instrukcji przypisywania lub samodzielnie napisanych funkcji i procedur. Zabronione jest używanie funkcji wbudowanych, dostępnych w językach programowania, a zwłaszcza funkcji podnoszącej do potęgi.

Specyfikacja algorytmu

Dane

k – dodatnia liczba całkowita, liczba rund w turnieju

x, *y* – dwie różne liczby całkowite z przedziału $[0, 2^k – 1]$, numery drużyn

Wynik

runda – nr rundy, w której mogą się spotkać drużyny *x* i *y*

Algorytm:

Wypełnia egzaminator	Nr zadania	2.1.	2.2.
	Maks. liczba pkt.	2	4
	Uzyskana liczba pkt.		

Zadanie 2. Analiza algorytmu

Wykonaj analizę funkcji *Algo*(*n*), której argumentem jest dodatnia liczba całkowita *n*.

```
Algo(n)
    jeżeli n ≤ 2 to
        wynikiem jest 1
    w przeciwnym przypadku
        p ← 1
        k ← n
        dopóki k – p > 1 wykonuj
            s ← (p + k) div 2
            jeżeli s * s ≤ n to
                p ← s
            w przeciwnym przypadku
                k ← s
        wynikiem jest p
```

Uwaga: *div* oznacza dzielenie całkowite.

Zadanie 2.1. (0–2)

Uzupełnij tabelę – podaj wynik funkcji *Algo* dla podanych w tabeli wartości *n*.

n	**Wynik otrzymany po wywołaniu *Algo*(*n*)**
5	2
35	
1025	

Miejsce na obliczenia:

Zadanie 2.2. (0–3)

Uzupełnij tabelę – podaj liczbę wykonań instrukcji „$s \leftarrow (p + k)$ *div* 2” podczas obliczania wartości funkcji *Algo*(*n*) dla podanych wartości *n.*

n	**Liczba wykonań instrukcji „$s \leftarrow (p + k)$ div 2” podczas obliczania wartości funkcji *Algo*(*n*)**
5	2
2	
63	
1024	

Miejsce na obliczenia

Wypełnia egzaminator	Nr zadania	2.1.	2.2.
	Maks. liczba pkt.	2	3
	Uzyskana liczba pkt.		

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)

W komórce C1 arkusza kalkulacyjnego zapisano formułę:

=JEŻELI(ORAZ(MOD(A1;2)=1;MOD(B1;2)=1);A1+B1;A1*B1)

1.	Jeśli w A1 wpisano liczbę 1, a w B1 liczbę 3, to w C1 w wyniku obliczenia formuły pojawi się liczba 4.	**P**	**F**
2.	Jeśli w A1 wpisano liczbę 4, a w B1 liczbę 3, to w C1 w wyniku obliczenia formuły pojawi się liczba 3.	**P**	**F**
3.	Jeśli w A1 i B1 wpiszemy dowolną liczbę całkowitą dodatnią, to w wyniku obliczenia formuły w C1 zawsze pojawi się liczba parzysta.	**P**	**F**
4.	Jeśli w A1 i B1 wpiszemy dowolną liczbę całkowitą dodatnią, to w wyniku obliczenia formuły w C1 zawsze pojawi się liczba większa niż 1.	**P**	**F**

Zadanie 3.2. (0–1)

Mamy dane operacje (bramki) logiczne na bitach: *not* oraz *and* opisane poniżej:

a	not *a*
1	0
0	1

a	*b*	*a* and *b*
1	1	1
0	1	0
1	0	0
0	0	0

oraz wyrażenie W(*a*,*b*):

(not ((not *a*) and *b*)) and (not (*a* and (not *b*)))

1.	W(0,0)=0	**P**	**F**
2.	W(1,0)=0	**P**	**F**
3.	W(0,1)=1	**P**	**F**
4.	W(1,1)=1	**P**	**F**

Zadanie 3.3. (0–1)

Różnica $1011101_2 - 10111_2$ dwóch liczb zapisanych w systemie binarnym jest:

1.	mniejsza niż 100111_2	**P**	**F**
2.	równa 1000110_2	**P**	**F**
3.	większa niż 10111_2	**P**	**F**
4.	równa 1001000_2	**P**	**F**

Zadanie 3.4. (0–1)

W bazie danych istnieje tabela *oceny(id_oceny, id_ucznia, przedmiot, ocena)*, zawierająca następujące dane:

id_oceny	id_ucznia	przedmiot	ocena
1	1	matematyka	3
2	1	informatyka	4
3	1	fizyka	2
4	2	matematyka	6
5	2	fizyka	3
6	2	informatyka	5
7	3	matematyka	4
8	3	fizyka	2
9	3	informatyka	3

1.	Wynikiem zapytania SELECT COUNT(id_ucznia) FROM oceny; jest 3	**P**	**F**
2.	Wynikiem zapytania SELECT COUNT (id_ucznia) FROM oceny WHERE przedmiot="fizyka"; jest 3	**P**	**F**
3.	Wynikiem zapytania SELECT COUNT(przedmiot) FROM oceny; jest 9	**P**	**F**
4.	Wynikiem zapytania SELECT COUNT(przedmiot) FROM oceny WHERE ocena > 3; jest 4	**P**	**F**

Wypełnia egzaminator	Nr zadania	3.1.	3.2.	3.3.	3.4.
	Maks. liczba pkt.	1	1	1	1
	Uzyskana liczba pkt.				

BRUDNOPIS ***(nie podlega ocenie)***