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:
Windows 10
(system operacyjny)
MS Office 2019
(program użytkowy)
PyCharm
(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.

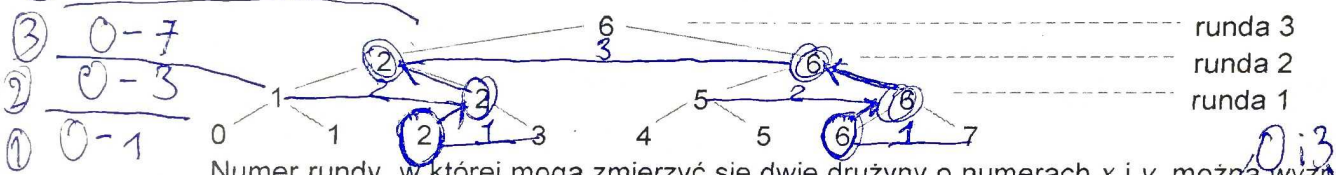
EINP-R1-**100**-2103

## 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 | 3 |
| 5 | 16 | 30 | 10000 | 11110 | 4 |

Miejsce na obliczenia:

$2 \cdot (i - 1)$ i $2 \cdot (i - 1) + 1 \longrightarrow i$ = numer meczu

dz

**Zadanie 1.2. (0–4)** ~ 10 minut

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:

```
jeśli x < y:
    max = y - x
w innym wypadku:
    max = x - y
counter = 1
num = 2
wykonuj k razy:
    jeśli num - 1 ≥ max:
        wypisz wartość counter i zakończ program
    w innym wypadku:
        num = num • 2
        counter = counter + 1
```

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

→ Estimating square root

**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
```

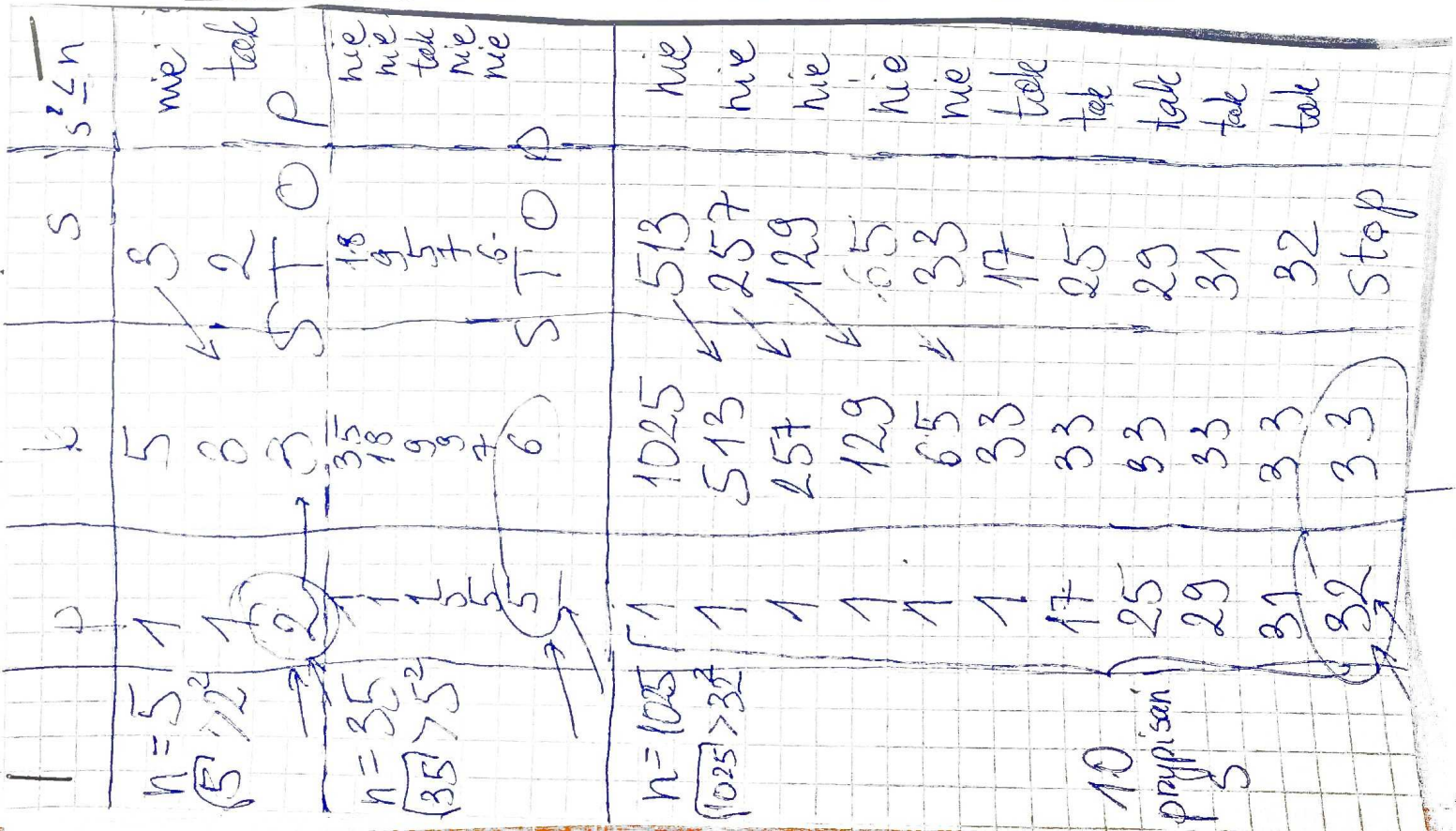
```
def Algo(n):
    if n ≤ 2:
        return 1
    else:
        p = 1
        k = n
        while k-p > 1:
            s = (p+k) // 2
            if: s*s ≤ n:
                p = s
            else:
                k = s
        return p
```

**Uwaga**: *div* oznacza dzielenie całkowite.

**Zadanie 2.1. (0–2)** ~7 minut

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

| *n* | Wynik otrzymany po wywołaniu *Algo*(*n*) |
|---|---|
| 5 | 2 |
| 35 | 5 |
| 1025 | 32 |



1/3

**Zadanie 2.2. (0–3)** ~~~3 minuty~~~

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 | 0 |
| 63 | ~~3~~ 6 |
| 1024 | ~~5~~ 10 |

Miejsce na obliczenia

| | p | k | s | $s^2 \leq n$ |
|---|---|---|---|---|
| n = 63 | 1 | 63 | 32 ① | nie |
| | 1 | 32 | 16 ② | nie |
| | 1 | 16 | 8 ③ | tak |
| | 8 | 16 | 12 ④ | nie |
| | 8 | 12 | 10 ⑤ | nie |
| | 8 | 10 | 9 ⑥ | nie |
| | 8 | 9 | — | — koniec |
| n = 1024 | 1 | 1024 | 512 ① | nie |
| | 1 | 512 | 256 ② | nie |
| | 1 | 256 | 128 ③ | nie |
| | 1 | 128 | 64 ④ | nie |
| | 1 | 64 | 32 ⑤ | tak |
| | 32 | 64 | 48 ⑥ | nie |
| | 32 | 48 | 40 ⑦ | nie |
| | 32 | 40 | 36 ⑧ | nie |
| | 32 | 36 | 34 ⑨ | nie |
| | 32 | 34 | 33 ⑩ | nie |
| | 32 | 33 | | koniec |

$2^1 = 2$
$2^2 = 4$
$2^3 = 8$
$2^4 = 16$
$2^5 = 32$
$2^6 = 64$
$2^7 = 128$
$2^8 = 256$
$2^9 = 512$
$2^{10} = 1024$

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

ok, całkowicie

4/4

## Zadanie 3. Test ~ 10 min

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.

1/1

### Zadanie 3.1. (0–1) ~ 2 min

W komórce C1 arkusza kalkulacyjnego zapisano formułę:

nieparz i nieparz — inne kombinacje

=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 |

1+3 → 4
2·3 → 6
2·2 → 4
1+1 = 2
2·1 = 2

1/1

### Zadanie 3.2. (0–1) ~ 3 min

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 |

$\sim(\sim p \wedge q) \wedge \sim(p \wedge \sim q)$
$\Longleftrightarrow$
$(p \vee \sim q) \wedge (\sim p \vee q)$

oraz wyrażenie W(*a*,*b*):

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

| | | | | |
|---|---|---|---|---|
| 1. | W(0,0)=0 | $(0 \vee 1) \wedge (1 \vee 0) \Longleftrightarrow 1$ | P | F |
| 2. | W(1,0)=0 | $(1 \vee 1) \wedge (0 \vee 0) \Longleftrightarrow 0$ | P | F |
| 3. | W(0,1)=1 | $(0 \vee 0) \wedge (0 \vee 1) \Longleftrightarrow 0$ | P | F |
| 4. | W(1,1)=1 | $(1 \vee 0) \wedge (0 \vee 1) \Longleftrightarrow 1$ | P | F |

1011101
− 10111
1000110 → 7 cyfr

111

**Zadanie 3.3. (0–1)** ~ 2 min

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

| | | | |
|---|---|---|---|
| 1. | mniejsza niż $100111_2$ → 6 cyfr | P | ~~F~~ |
| 2. | równa $1000110_2$ | ~~P~~ | F |
| 3. | większa niż $10111_2$ → 5 cyfr | ~~P~~ | F |
| 4. | równa $1001000_2$ | P | ~~F~~ |

111

**Zadanie 3.4. (0–1)** ~ 2 min

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 |

brak grupowania, liczy się każdy → 9

| | | | |
|---|---|---|---|
| 1. | Wynikiem zapytania<br>SELECT COUNT(id_ucznia) FROM oceny;<br>jest<br>3 | P | ~~F~~ |
| 2. | Wynikiem zapytania<br>SELECT COUNT (id_ucznia) FROM oceny<br>WHERE przedmiot="fizyka";<br>jest<br>3 | ~~P~~ | F |
| 3. | Wynikiem zapytania<br>SELECT COUNT(przedmiot) FROM oceny;<br>jest<br>9 → ok, liczy się każdy | ~~P~~ | F |
| 4. | Wynikiem zapytania<br>SELECT COUNT(przedmiot) FROM oceny<br>WHERE ocena > 3;<br>jest<br>4 | ~~P~~ | F |

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