**Arkusz I 2022 - Klucz rozwiązań**

# Zadanie 1.1 (0-1) Test

**Poprawna odpowiedź: F,F,P,F**

# Zadanie 1.2 (0-1)

**Poprawna odpowiedź: P, F, F, P**

# Zadanie 1.3. (0–1)

**Poprawna odpowiedź: F, P, F, F**

# Zadanie 1.4. (0–1)

**Poprawna odpowiedź: F, F, F, P**

# Zadanie 1.5. (0–1)

**Poprawna odpowiedź: P, P, P, P.**

# Zadanie 2. (0-5) Liczby dwupierwsze

## Zadanie 2.1. (0–1)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 𝑝 |  | 𝑞 |  | ̅𝑝𝑞̅̅ |
| 61 | 67 |  | 6671 |  |
| 83 | 19 |  | 8139 |  |
| 7 | 71 |  | 771 |  |

## Zadanie 2.2. (0–2)

**C++:**

#include <iostream>

**using** **namespace** std;

**int** main() {

**int** jednocyfrowe[4] = {2, 3, 5, 7};

**int** dwucyfrowe[21] = {11, 13, 17, 19, 23, 29, 31, 37, 41, 43,

47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97}; **int** n; **for**(**int** i = 0; i < 4; i++)

{

**for**(**int** j = 0; j < 21; j++)

{

n = 100\*(dwucyfrowe[j]/10) + 10\*jednocyfrowe[i] +

(dwucyfrowe[j]%10);

**cout** << n << **endl**;

}

}

**return** 0; }

**Python:**

jednocyfrowe = [2, 3, 5, 7]

dwucyfrowe = [11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59,

61, 67, 71, 73, 79, 83, 89, 97]

**for** p **in** jednocyfrowe: **for** q **in** dwucyfrowe:

n = 100\*(q//10) + 10\*p + (q%10) **print**(n)

**Lista kroków:**

**Krok 1.** jednocyfrowe := [2, 3, 5, 7]

**Krok 2.** dwucyfrowe := [11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]

**Krok 3.** Dla wszystkich liczb p w tablicy jednocyfrowe wykonuj Kroki 4-6

**Krok 4.** Dla wszystkich liczb q w tablicy dwucyfrowe wykonuj Kroki 5-6

**Krok 5.** n := n = 100\*(q div 10) + 10\*p + (q mod 10) **Krok 6.** Wypisz n

Uwaga. Uczeń zamiast wypisywać liczby pierwsze może je wygenerować algorytmem, np. sitem Eratostenesa, lub przebiec pętlami po wszystkich liczbach jednocyfrowych i dwucyfrowych, sprawdzać, czy liczby są pierwsze i dopiero wtedy generować liczbę dwupierwszą.

**Punktacja:**

2 pkt – rozwiązanie w pełni poprawne

1 pkt – za propozycję algorytmu wypisującego wszystkie takie liczby, lecz z powtórzeniami; lub za poprawny algorytm, lecz z usterką w tablicach z liczbami pierwszymi; lub za poprawny algorytm, lecz z usterką w algorytmie wyznaczającym kolejne liczby pierwsze 0 pkt – w pozostałych przypadkach

## Zadanie 2.3. (0–2)

C++:

#include <iostream>

**using** **namespace** std;

**void** Rozloz(**unsigned** **int** n, **unsigned** **int**& p, **unsigned** **int**& q)

{

**int** k = 1, r;

p = 0; q = 0; **while**(n > 0) { r = n%10; p += r\*k; n = n/10; r = n%10; q += r\*k; n = n/10; k \*= 10;

}

}

**bool** CzyPierwsza(**unsigned** **int** n)

{ **int** d = 2; **if**(n == 1) **return** **false**;

**if**(n == 2 || n == 3) **return** **true**;

**while**(d\*d <= n) { **if**(n % d == 0) **return** **false**; **else** d++;

} **return** **true**;

} **int** **main**() {

**unsigned** **int** n, p, q; **cin** >> n;

Rozloz(n, p, q);

**if**(CzyPierwsza(p) && CzyPierwsza(q)) **cout** << "TAK" << **endl**; **else**

**cout** << "NIE" << **endl**;

**return** 0;

}

Python:

**def** Rozloz(n): k = 1

p = q = 0 **while** n > 0: r = n%10 p += r\*k n = n//10 r = n%10 q += r\*k n = n//10 k \*= 10 **return** p, q **def** CzyPierwsza(n):

d = 2 **if** n==1:

**return** **False**

**if** n==2 **or** n==3: **return** **True**

**while** d\*d<=n: **if** n%d == 0: **return** **False** **else**: d += 1

**return** **True** n = **int**(**input**("Podaj n: ")) p, q = Rozloz(n) **if** CzyPierwsza(p) **and** CzyPierwsza(q):

**print**("TAK") **else**: **print**("NIE")

Lista kroków:

**Krok 1.** k := 0, p := 0, q := 0

**Krok 2.** Dopóki n > 0 wykonuj Kroki 3-9:

**Krok 3.** r := n mod 10

**Krok 4.** p := p + r\*k

**Krok 5.** n := n div 10

**Krok 6.** r := n mod 10

**Krok 7.** q := q + r\*k

**Krok 8.** n := n div 10

**Krok 9.** k := k\*10

**Krok 10.** d := 2

**Krok 11.** Jeżeli p = 1 lub q = 1, to wypisz NIE i zakończ algorytm

**Krok 12.** Jeżeli p = 2 lub p = 3 to idź do Kroku 17

**Krok 13.** Dopóki d\*d <= p wykonuj Kroki 14-15

**Krok 14.** Jeżeli p mod d = 0, to wypisz NIE i zakończ algorytm

**Krok 15.** d := d + 1

**Krok 16.** d := 2

**Krok 17.** Jeżeli q = 2 lub q = 3 to wypisz TAK i zakończ algorytm

**Krok 18.** Dopóki d\*d <= q wykonuj Kroki 19-20

**Krok 19.** Jeżeli q mod d = 0, to wypisz NIE i zakończ algorytm

**Krok 20.** d := d + 1

**Krok 21.** Wypisz TAK i zakończ algorytm

# Zadanie 3. (0-5) FlipSort

|  |  |
| --- | --- |
| Nr pytania | **Oczekiwana odpowiedź** |
| 3.1 | |  |  |  | | --- | --- | --- | | **tab** | **flip(a,b)** | **Tablica po wykonaniu operacji flip** | | [1,4,8,3,8,2,9] | flip(2,6) | [1,2,8,3,8,4,9] | | [1,2,7,6,3,9,0] | flip(3,5) | [1,2,3,6,7,9,0] | | [1,2,3,7,9,4,5] | flip(4,6) | [1,2,3,4,9,7,5] | |
| 3.2 | |  |  | | --- | --- | | **Tablica tab** | **Sekwencja operacji flip** | | [2, 3, 1, 7, 5] | flip(1,3), flip(2,3), flip(4,5) | | [4, 7, 1, 2, 3] | flip(1,3), flip(2,4), flip(3,5), flip(4,5) | |
| 3.3 | **Odpowiedź: n-1** |
| 3.4 | **Przykładowe rozwiązanie:**    funkcja szukajMin(n, tab, ind):  min := tab[ind] minInd := ind od i := ind do n, wykonuj: |
|  | jeżeli tab[i] < min, to:  min := tab[i] minInd := i zwróć minInd  funkcja flipSort(n, tab): Od i := 1 do n – 1, wykonuj:  minInd = szukajMin(n, tab, i) jeżeli minInd != i, to:  flip(i, minInd)  wypisz „flip(„ + i + „,” + minInd  + „)”  flipSort(n, tab) |