Arkusz I 2022 - Klucz rozwiązań

Zadanie 1.1 (0-1) Test

Poprawna odpowiedź: F,F,P,F

Wymagania ogólne	Wymagania szczegółowe
III. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, z zastosowaniem podejścia algorytmicznego.	Zdający analizuje i rozwiązuje sytuacje problemowe z różnych dziedzin (5.1), Zdający stosuje podejście algorytmiczne do rozwiązywania problemu (5.2), stosuje rekurencję w prostych sytuacjach problemowych (5.9), opisuje własności algorytmów na podstawie ich analizy (16)

Zadanie 1.2 (0-1)

Poprawna odpowiedź: P, F, F, P

Wymagania ogólne	Wymagania szczegółowe
V. Ocena zagrożeń i ograniczeń, docenianie społecznych aspektów rozwoju i zastosowań informatyki.	Uczeń Komunikuje się za pomocą komputera i technologii informacyjno-komunikacyjnych. (3) Uczeń wykorzystuje komputer i technologie informacyjno-komunikacyjne do rozwijania swoich zainteresowań, opisuje zastosowania informatyki, ocenia zagrożenia i ograniczenia (7), stosuje normy etyczne i prawne związane z ochroną danych oraz informacji w komputerze i sieciach komputerowych. (7.3)

Zadanie 1.3. (0–1)

Poprawna odpowiedź: F, P, F, F

Wymagania ogólne	Wymagania szczegółowe
III. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, z zastosowaniem podejścia algorytmicznego.	Zdający analizuje i rozwiązuje sytuacje problemowe z różnych dziedzin (5.1), Zdający stosuje podejście algorytmiczne do rozwiązywania problemu (5.2), dobiera odpowiednie struktury danych do realizacji algorytmu, w tym struktury dynamiczne (14)

Zadanie 1.4. (0–1)

Poprawna odpowiedź: F, F, F, P

Wymagania ogólne	Wymagania szczegółowe
I. Bezpieczne posługiwanie się komputerem i jego oprogramowaniem, wykorzystanie sieci komputerowej; komunikowanie się za pomocą komputera i technologii informacyjno-komunikacyjnych.	Zdający przedstawia sposoby reprezentowania różnych form informacji w komputerze: liczb, znaków, obrazów, animacji, dźwięków (1.1, A 1.1).

Zadanie 1.5. (0–1)

Poprawna odpowiedź: P, P, P, P.

Wymagania ogólne	Wymagania szczegółowe
II. Wyszukiwanie, gromadzenie i przetwarzanie informacji z różnych źródeł […].	Zdający wyszukuje, gromadzi, selekcjonuje, przetwarza informacje […] (2.). Stosuje metody wyszukiwania i przetwarzania informacji w relacyjnej bazie danych (2.2)

Zadanie 2. (0-5) Liczby dwupierwsze

Wymagania ogólne	Wymagania szczegółowe
III. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego.	Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego. (5, A 4) Zdający analizuje, modeluje i rozwiązuje sytuacje problemowe z różnych dziedzin (5.1, A 4.1), stosuje algorytmiczne podejście do rozwiązywania problemu (5.2, A 4.2), dobiera efektywne algorytm do rozwiązania sytuacji problemowej i zapisuje go w wybranej notacji (5.4, A 4.4)

Zadanie 2.1. (0–1)

p	q	$\overline{pq}$
61	67	6671
83	19	8139
7	71	771

Punktacja:

1 pkt – poprawnie wypełniona tabela
0 pkt – w pozostałych przypadkach

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

Punktacja:

2 pkt – rozwiązanie w pełni poprawne
1 pkt – za poprawny rozkład n na p i q oraz niepoprawne sprawdzenie pierwszości p lub q (lub jego brak); lub za usterkę w rozkładzie n na p i q (lub jego brak) oraz poprawne sprawdzenie pierwszości p i q
0 pkt – w pozostałych przypadkach

Zadanie 3. (0-5) FlipSort

Wymagania ogólne	Wymagania szczegółowe
III. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego.	5. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego. **Zdający:** 1) analizuje, modeluje i rozwiązuje sytuacje problemowe z różnych dziedzin; 2) stosuje algorytmiczne podejście do rozwiązywania problemu; 4) dobiera efektywne algorytm do rozwiązania sytuacji problemowej i zapisuje go w wybranej notacji;

<table>
<tr><th>Nr pytania</th><th>Oczekiwana odpowiedź</th><th colspan="2">Maksymalna punktacja</th></tr>
<tr><td>3.1</td><td>Za podanie pełnej poprawnej odpowiedzi – 1 punkt.
Odpowiedź:
<table>
<tr><th>tab</th><th>flip(a,b)</th><th>Tablica po wykonaniu operacji flip</th></tr>
<tr><td>[1,4,8,3,8,2,9]</td><td>flip(2,6)</td><td>[1,2,8,3,8,4,9]</td></tr>
<tr><td>[1,2,7,6,3,9,0]</td><td>flip(3,5)</td><td>[1,2,3,6,7,9,0]</td></tr>
<tr><td>[1,2,3,7,9,4,5]</td><td>flip(4,6)</td><td>[1,2,3,4,9,7,5]</td></tr>
</table></td><td>1</td><td rowspan="4">5</td></tr>
<tr><td>3.2</td><td>Za podanie pełnej poprawnej odpowiedzi: 1 punkt.
Odpowiedź:
<table>
<tr><th>Tablica tab</th><th>Sekwencja operacji flip</th></tr>
<tr><td>[2, 3, 1, 7, 5]</td><td>flip(1,3), flip(2,3), flip(4,5)</td></tr>
<tr><td>[4, 7, 1, 2, 3]</td><td>flip(1,3), flip(2,4), flip(3,5), flip(4,5)</td></tr>
</table></td><td>1</td></tr>
<tr><td>3.3</td><td>Za podanie pełnej poprawnej odpowiedzi: 1 punkt.
Odpowiedź: n-1</td><td>1</td></tr>
<tr><td>3.4</td><td>Za podanie poprawnego algorytmu: 2 punkty.
W tym 1 punkt za poprawną implementację znajdowania wartości minimalnej w podanym zakresie tablicy.

Przykładowe rozwiązanie:

<pre>funkcja szukajMin(n, tab, ind):
    min := tab[ind]
    minInd := ind
    od i := ind do n, wykonuj:</pre></td><td>2</td></tr>
</table>

	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)		