Omówienie Arkusza Matury z Informatyki Maj 2019

Kacper Skrzypiec 3c2

Zadanie 1. Ulubione liczby

Małgosia i Jaś lubią liczby. Małgosia lubi liczby nieparzyste, a Jaś lubi liczby parzyste. Każde z dzieci zapisało po kilka spośród swoich ulubionych liczb na jednej wspólnej kartce. Najpierw Małgosia zapisała wszystkie swoje liczby, a potem Jaś dopisał swoje.

Zadanie 1.1. (0–5)

Napisz algorytm (w postaci listy kroków, w pseudokodzie lub w wybranym języku programowania), który dla danego ciągu liczb zapisanych przez dzieci znajdzie pierwszą liczbę zapisaną przez Jasia. Zakładamy, że każde z dzieci zapisało co najmniej jedną liczbę.

Przy ocenie będzie brana pod uwagę złożoność czasowa Twojego algorytmu. Maksymalną liczbę punktów uzyskasz za algorytm o złożoności lepszej niż liniowa.

Uwaga: W zapisie algorytmu możesz wykorzystać tylko operacje arytmetyczne (dodawanie, odejmowanie, mnożenie, dzielenie, dzielenie całkowite, reszta z dzielenia), instrukcje porównania, instrukcje sterujące i przypisania do zmiennych lub samodzielnie napisane funkcje, wykorzystujące wyżej wymienione operacje.

Specyfikacja:

Dane:

n – liczba całkowita większa od 1

$A[1..n]$ – tablica zawierająca ciąg n liczb zapisanych przez dzieci (najpierw wszystkie liczby nieparzyste, a potem wszystkie liczby parzyste)

Wynik:

w – pierwsza od lewej parzysta liczba w tablicy A

Przykład:

Dane:

$n = 10$

$A[1..n] = \{5, 99, 3, 7, 111, 13, 4, 24, 4, 8\}$

Wynik:

$w = 4$

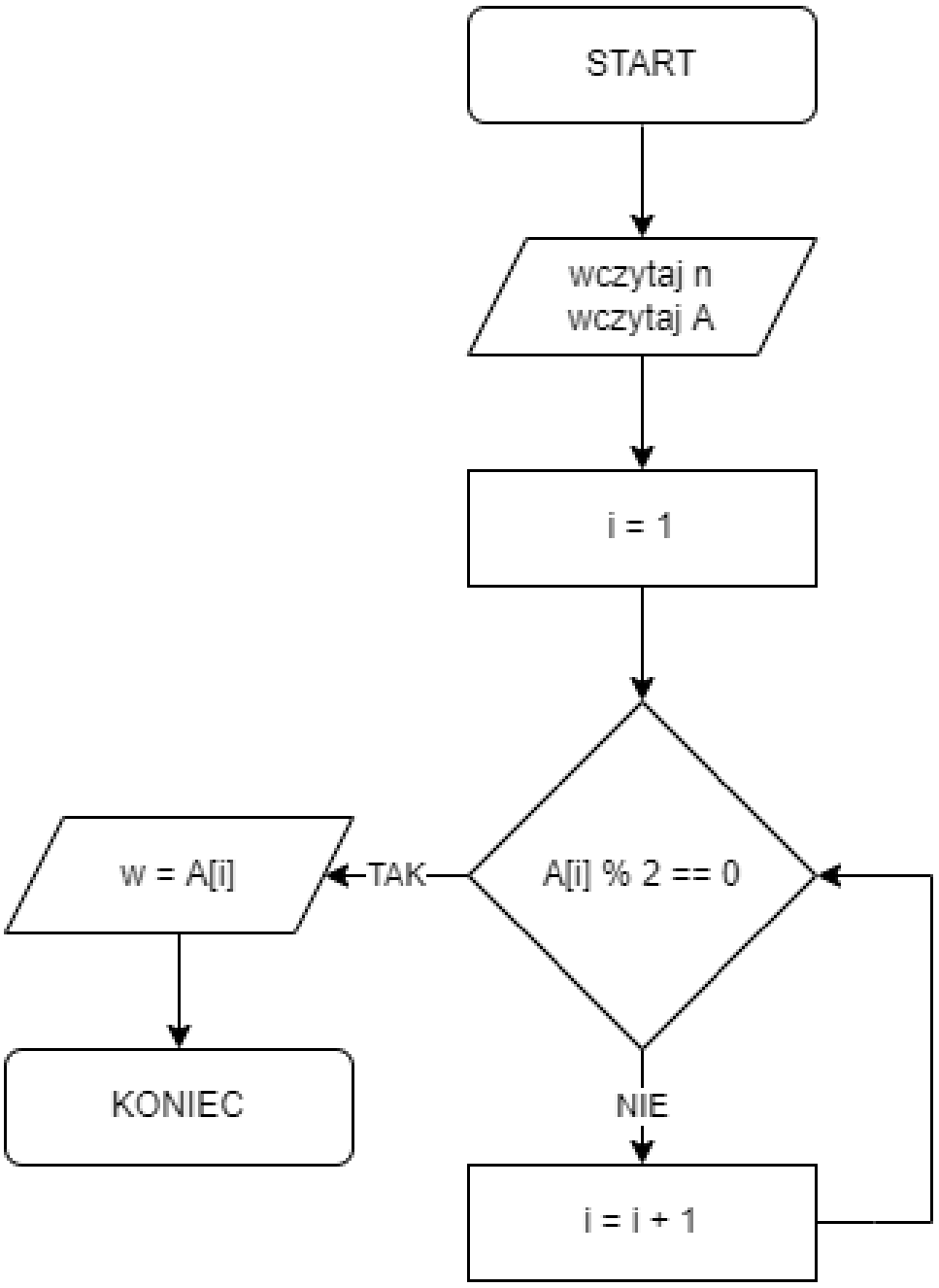
Zadanie 1.2. (0–1)

Podaj, jaką złożoność czasową – kwadratową, liniową, logarytmiczną lub inną (napisz jaką) – ma Twój algorytm.

.....

Przedstawione powyżej zadanie można rozwiązać na dwa sposoby różniące się złożonością czasową.

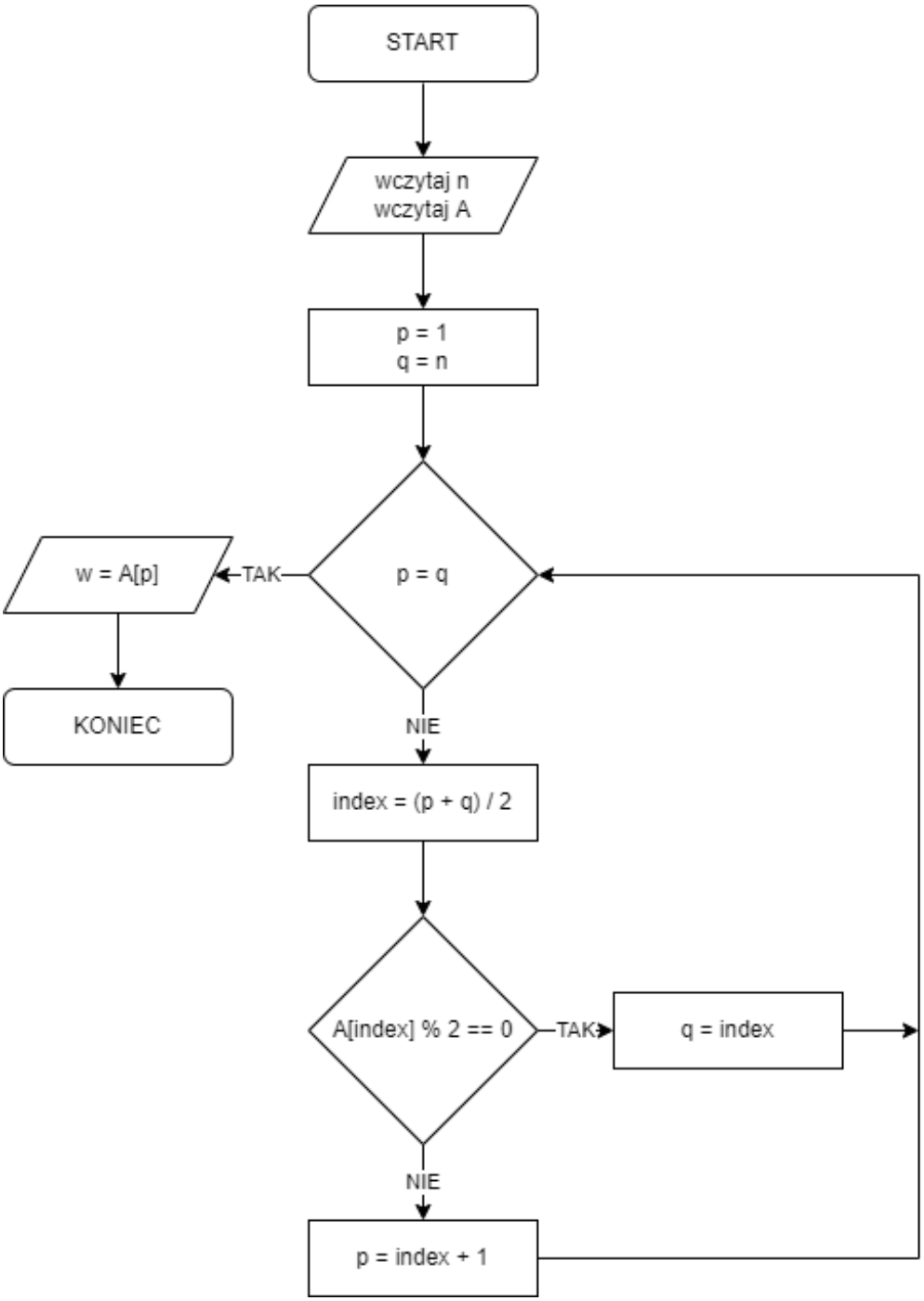
Pierwszy algorytm polega na przeszukiwaniu tablicy element po elemencie i sprawdzaniu czy jest liczbą parzystą. Jeżeli tak to zwróci jej indeks i zakończy pętlę.



Złożoność przedstawionego rozwiązania jest liniowa:

$$O(n)$$

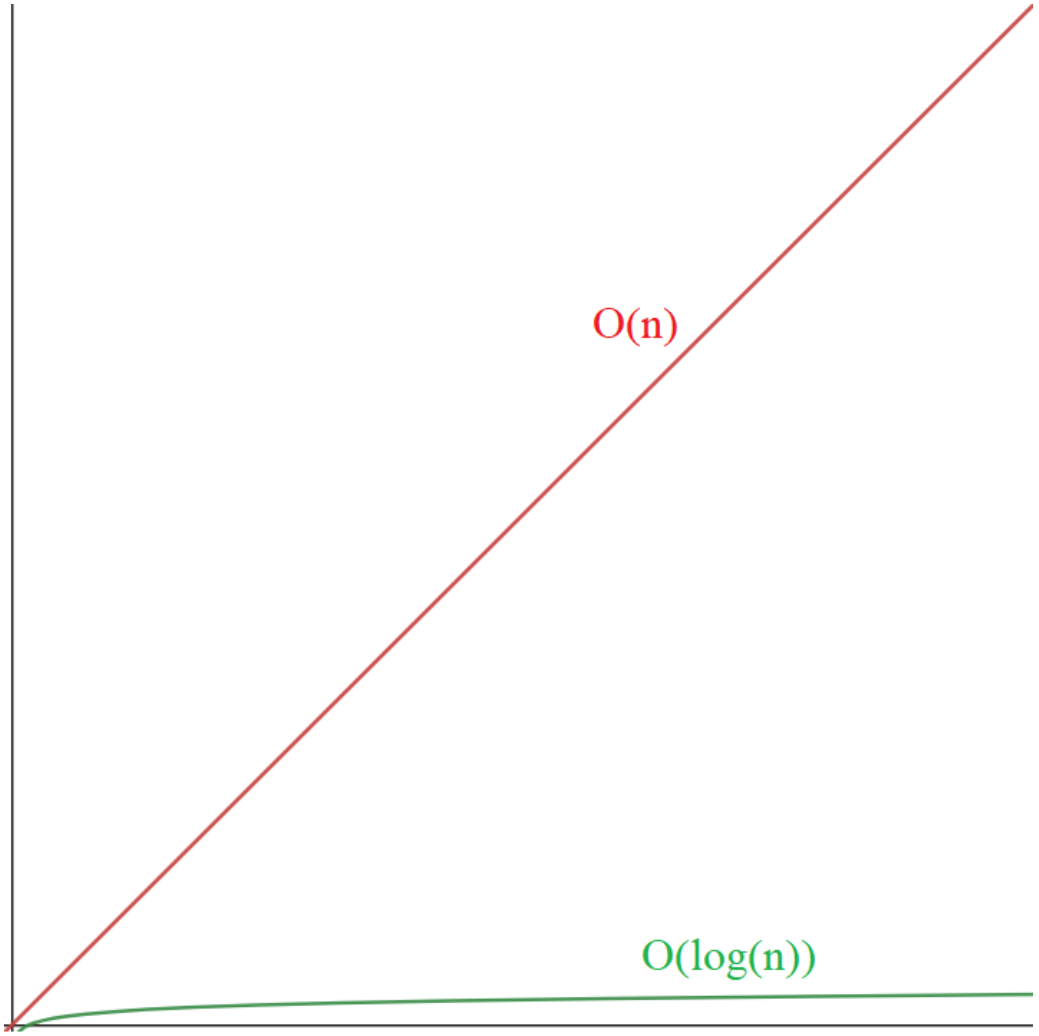
Drugi algorytm stosuje *wyszukiwanie binarne*, które opiera się na metodzie dziel i zwyciężaj. Film wyjaśniający jego działanie: Binary Search Algorithm in 100 Seconds.



Jego złożoność czasowa jest logarytmiczna:

$$O(log(n))$$

Poniżej znajduje się wykres porównujący obie złożoności:



Jak można zauważyć, różnica w wydajności jest znacząca.

Zadanie 2. Analiza algorytmu

Przeanalizuj podaną funkcję `pisz`.

Specyfikacja:

Dane:

- s – napis
- n – liczba całkowita dodatnia, nie mniejsza niż długość napisu s
- k – liczba całkowita z zakresu [2..10]

```
funkcja pisz(s,n,k)
     jeżeli dł(s) = n
         wypisz s
     w przeciwnym razie
          dla i=0,1 … k-1 wykonuj
               pisz(s + napis(i), n, k)
```

Uwaga:

dł(x) – daje w wyniku długość napisu x

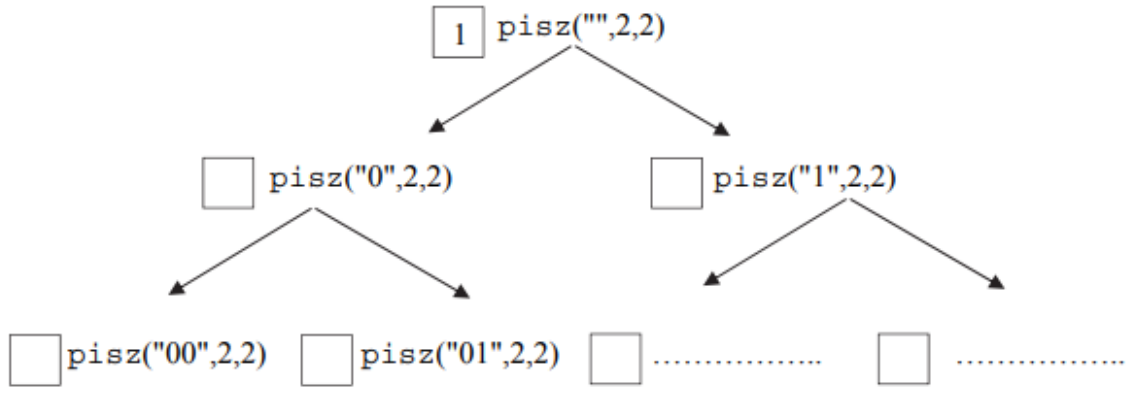
$s1 + s2$ – daje w wyniku złączenie napisów $s1$ i $s2$

napis(p) – daje w wyniku napis będący zapisem dziesiętnym liczby całkowitej p

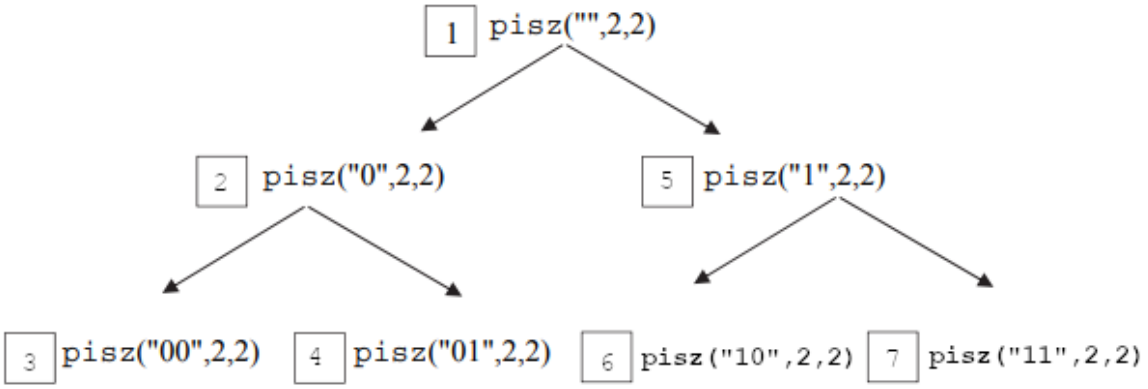
Zadanie 2.1. (0–2)

a) Uzupełnij miejsca oznaczone kropkami w drzewie wywołań funkcji `pisz` otrzymanym w wyniku wywołania `pisz("",2,2)`.

b) W kwadratowych polach, przy węzłach drzewa, podaj odpowiednią kolejność wywołań funkcji `pisz`, tzn. przy pierwszym wywołaniu – 1, przy kolejnym – 2 itd.



Zadaniem funkcji w powyższym zadaniu jest wyświetlać podany napis **s** jeżeli dana długość jest równa jego długości. W przeciwnym razie funkcja wywoła się **k** razy wewnątrz siebie i tyle też powstanie nowych napisów, do których po kolei przyłączy się cyfra(według założeń będą to tylko cyfry) od **0** do **k - 1**. Argumenty **n** i **k** nie ulegają zmianie.

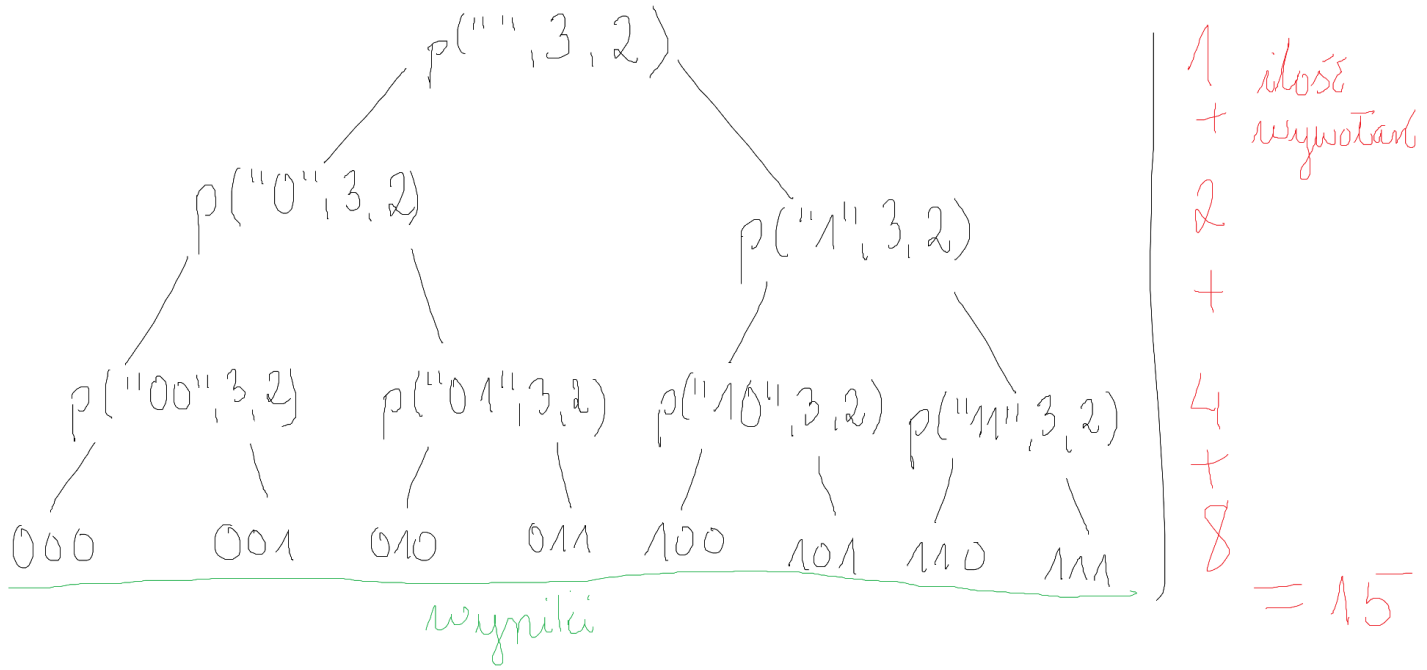


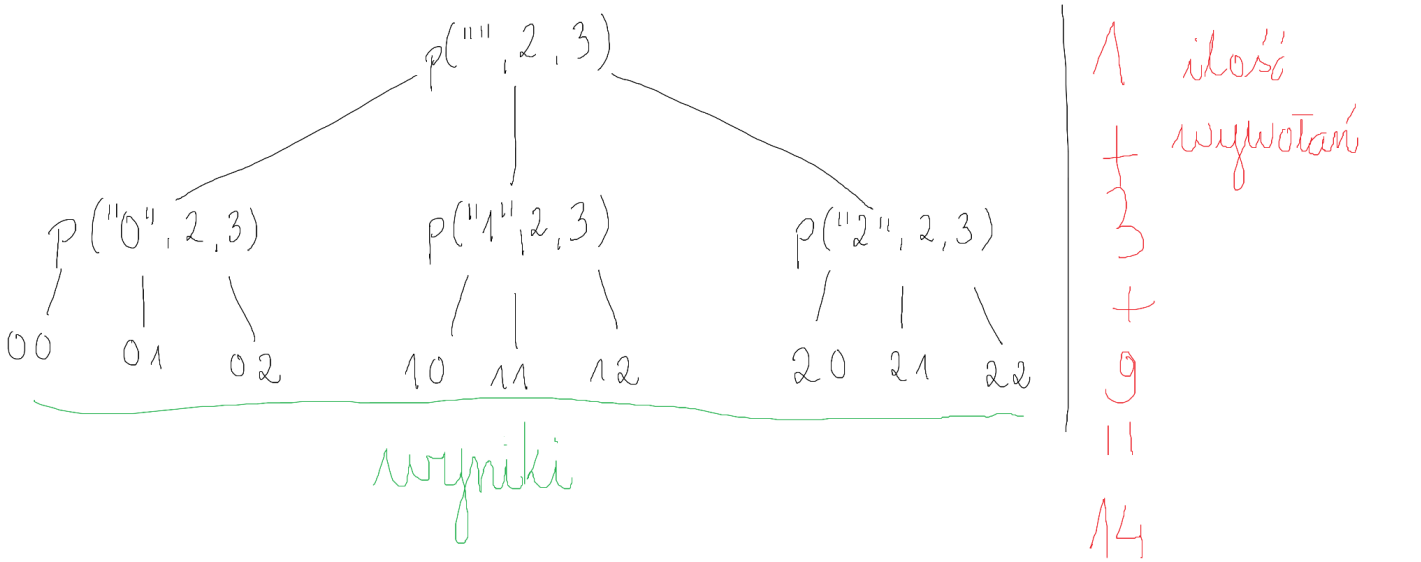
Aby wypełnić kolejność wywoływania funkcji należy prześledzić jej działanie, kiedy wywołuje samą siebie. Jak widać, **pisz(“0”, 2, 2)** będzie drugim wywołaniem. Następnie wywoła **pisz(“00”, 2, 2)** i **pisz(“01”, 2, 2)**, dalej nic nie stanie, bo osiągnięto warunek **dł(s) = n**, więc powrócimy do pisz(“”, 2, 2) i zaczniemy wywoływać **pisz(“1”, 2, 2)** i dalej analogicznie.

Zadanie 2.2. (0–2)

Uzupełnij poniższą tabelę – przeanalizuj podane w niej wywołania funkcji `pisz`. Podaj napisy wypisywane w wyniku wywołania funkcji `pisz` z zadanymi parametrami oraz łączną liczbę wywołań tej funkcji.

Pierwsze wywołanie funkcji `pisz`	Napisy wypisane w wyniku wywołania funkcji `pisz`	Łączna liczba wywołań funkcji `pisz`
`pisz("", 3, 2)`		
`pisz("", 2, 3)`		


p("",3,2)
p("0",3,2)
p("1",3,2)
p("00",3,2)
p("01",3,2)
p("10",3,2)
p("11",3,2)
000
001
010
011
100
101
110
111
wyniki
1 ilość wywołań
+
2
+
4
+
8
= 15


p("",2,3)
p("0",2,3)
p("1",2,3)
p("2",2,3)
00
01
02
10
11
12
20
21
22
wyniki
1 ilość wywołań
+
3
+
9
11
14

Zadanie 2.3. (0–2)

Podaj wzór na łączną liczbę wywołań funkcji `pisz` w wyniku wywołania `pisz("", n, k)`.

.....

Po przerobieniu poprzedniego podpunktu bez trudu można zauważyć, że mamy do czynienia z ciągiem geometrycznym.

n+1 odpowiada ilości poziomów(przez poziom rozumiem sekcje poziomą z funkcjami), a **k** ilości gałęzi od każdego wywołania, więc na każdym poziomie będzie k^p wywołań, gdzie p to dany poziom 0, 1, 2, …

Sumujemy ilość wywołań na każdym poziomie:

$$S = \sum_{i=0}^{n+1} k^i$$

Stosując wzór na sumę w ciągu geometrycznym (lub po jego zapomnieniu dokonując przekształceń zaczynając od pomnożenia S przez -k) otrzymujemy:

$$S = a_0 \frac{1-q^N}{1-q} = k^0 \frac{1-k^{n+1}}{1-k} = \frac{1-k^{n+1}}{1-k}$$

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)

Dana jest tabela `PRACOWNICY`.

Nr_P	Nazwisko	Imię	Stanowisko	Nr_działu
736	Smitko	Alan	urzędnik	20
7499	Nowak	Kazimierz	sprzedawca	30
7521	Więcek	Mariusz	sprzedawca	30
7566	Jonas	Kamil	kierownik	20
7654	Martin	Leon	sprzedawca	30
7698	Bracki	Bartosz	kierownik	30
7782	Celerek	Agnieszka	kierownik	10
7788	Skotnik	Natalia	analityk	20
7839	King	Mirosława	prezes	10

1.	Wynikiem zapytania **SELECT COUNT**(`Stanowisko`) **FROM** `PRACOWNICY`; jest Stanowisko 5	**P**	**F**
2.	Wynikiem zapytania **SELECT COUNT**(`Stanowisko`) **FROM** `PRACOWNICY` **WHERE** `Stanowisko` <> "kierownik"; jest 6	**P**	**F**
3.	Wynikiem zapytania **SELECT** `Stanowisko`, **COUNT**(*) **FROM** `PRACOWNICY` **GROUP BY** `Stanowisko`; jest urzędnik 1 sprzedawca 3 kierownik 3 analityk 1 prezes 1	**P**	**F**
4.	Wynikiem zapytania **SELECT COUNT**(`Stanowisko`) **FROM** `PRACOWNICY` **WHERE** `Stanowisko` **LIKE** "*nik"; jest 2	**P**	**F**

1)

Definition and Usage

The COUNT() function returns the number of records returned by a select query.

Note: NULL values are not counted.

Z definicji zwrócona wartość kwerendy powinna wynosić 9 po ręcznym przeliczeniu, a w przykładzie jest 5, więc jest to **FAŁSZ**

2)

The SQL WHERE Clause

The `WHERE` clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

Oznacza to, że policzymy (count) tylko rekordy, w których stanowisko jest różne od “kierownik”. Więc odpowiedzią będzie 6, co jest zgodne z przykładem, **PRAWDA**

3)

The SQL GROUP BY Statement

The `GROUP BY` statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

Ponownie, po przeliczeniu według definicji rezultat będzie taki jak w przykładzie, **PRAWDA**

4)

The SQL LIKE Operator

The `LIKE` operator is used in a `WHERE` clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the `LIKE` operator:

- The percent sign (%) represents zero, one, or multiple characters
- The underscore sign (_) represents one, single character

Note: MS Access uses an asterisk (*) instead of the percent sign (%), and a question mark (?) instead of the underscore (_).

The percent sign and the underscore can also be used in combinations!

Liczmy rekordy, w których ostatnie trzy literki stanowiska to “nik”. Widzimy z tabeli, że jest ich 5, a przykładzie wynikiem jest 2, więc **FAŁSZ**

źródło: https://www.w3schools.com/sql/

Zadanie 3.2. (0–1)
Po pomnożeniu dwóch liczb 1111110_2 oraz 101_2 zapisanych w systemie dwójkowym otrzymamy:

1.	21312_4	**P**	**F**
2.	1001010110_2	**P**	**F**
3.	1166_8	**P**	**F**
4.	276_{16}	**P**	**F**

Tutaj raczej nie ma nad czym się rozwodzić. Proste systemy liczbowe.

$$1111110_2 = 2^7 - 2 = 126_{10}$$
$$101_2 = 5_{10}$$
$$126 * 5 = 630$$

Można też od razu pomnożyć w kodzie binarnym(111111000 + 1111110), ale jest zbyt dużo jedynek i oczopląsu można dostać od tego.

Zmieniamy reprezentację liczby 630 z dziesiętnej na dwójkową:

630 | 0
315 | 1
157 | 1
78 | 0
39 | 1
19 | 1
9 | 1
4 | 0
2 | 0
1 | 1
0 —> 1001110110

0010 0111 0110 = 2 7 6 (16)
001 001 110 110 = 1 1 6 6 (8)
10 01 11 01 10 = 2 1 3 1 2 (4)

Więc odpowiedź to: **PFPP**

Zadanie 3.3. (0–1)

1.	DNS to skrót od Domain Name System.	**P**	**F**
2.	Do danego adresu IP może być przypisanych wiele różnych nazw.	**P**	**F**
3.	Przy zmianie adresu IP komputera pełniącego funkcję serwera WWW jest konieczna zmiana nazwy domeny internetowej.	**P**	**F**
4.	System DNS ma jedną centralną bazę danych adresów IP i nazw.	**P**	**F**

1)

PRAWDA

2)

PRAWDA,

"Multiple hostnames may correspond to a single IP address [...]"

~wikipedia 30.03.2022

3)

FAŁSZ,

DNS zmienia nazwę mnemoniczną na docelowy adres IP. Przy zmianie adresu IP serwera wystarczy tylko edytować przypisany adres IP w ustawieniach domeny

4)

FAŁSZ,

"The **Domain Name System** (**DNS**) is the hierarchical and ***decentralized*** naming system used to identify computers, services, and other resources reachable through the Internet or other Internet Protocol (IP) networks."

~wikipedia 30.03.2022