

1. Zadania rozwiązywane bez użycia komputera

1.1. Analiza algorytmów

Zadanie 1.

Wiązka zadań *Ciągi rekurencyjne*

Dana jest następująca funkcja rekurencyjna:

funkcja $wynik(i)$

jeżeli $i < 3$

zwróć 1 i zakończ;

w przeciwnym razie

jeżeli $i \bmod 2 = 0$

zwróć $wynik(i - 3) + wynik(i - 1) + 1$

w przeciwnym razie

zwróć $wynik(i - 1) \bmod 7$

Uwaga: Operator \bmod oznacza resztę z dzielenia.

1.1.

Uzupełnij poniższą tabelę:

<i>i</i>	wynik(<i>i</i>)
2	1
3	
4	
5	
6	
7	
8	

1.2.

Wykonaniem elementarnym nazywać będziemy wykonanie $wynik(0)$, $wynik(1)$ lub $wynik(2)$. Natomiast *złożonością elementarną* $wynik(i)$ nazywamy liczbę wykonień elementarnych będących efektem uruchomienia $wynik(i)$. Złożoność elementarną $wynik(i)$ oznaczamy przez $E(i)$.

Na przykład złożoność elementarna $wynik(4)$ wynosi $E(4) = 2$, ponieważ wykonując $wynik(4)$, wywołamy $wynik(3)$ i $wynik(1)$ (wykonanie elementarne), a z kolei przy wykonaniu $wynik(3)$ wywołamy $wynik(2)$ (drugie wykonanie elementarne).

Uzupełnij poniższą tabelę:

<i>i</i>	<i>E(i)</i>
0	1
3	1
5	
7	
9	
10	

Okazuje się, że $E(i)$ można opisać rekurencyjnym wyrażeniem, którego niekompletną postać podajemy poniżej. Uzupełnij brakujące miejsca tak, aby $E(i)$ dawało poprawną złożoność elementarną $wynik(i)$ dla każdego całkowitego nieujemnego i .

$$\begin{aligned} E(0) &= E(1) = E(2) = 1 \\ E(i) &= E(\dots) + E(\dots) \quad \text{dla parzystego } i > 2 \\ E(i) &= E(\dots) \quad \text{dla nieparzystego } i > 2 \end{aligned}$$

1.3.

Naszym celem jest wyznaczenie największej liczby spośród wartości funkcji $wynik(0), wynik(1), \dots, wynik(1000)$ bez konieczności rekurencyjnego wyznaczania kolejnych wartości. Poniżej prezentujemy niekompletny algorytm realizujący to zadanie.

```

 $W[0] \leftarrow 1$ 
 $W[1] \leftarrow 1$ 
 $W[2] \leftarrow 1$ 
 $max\_wart \leftarrow 1$ 
dla  $i = 3, 4, \dots, 1\,000$  wykonuj
    jeżeli  $i \bmod 2 = 0$ 
         $W[i] \leftarrow \dots$ 
    w przeciwnym razie
         $W[i] \leftarrow \dots$ 
    jeżeli  $W[i] > max\_wart$ 
         $\dots$ 
zwróć  $max\_wart$ 

```

Uzupełnij brakujące miejsca w algorytmie tak, aby zwracał on największą liczbę spośród $wynik(0), wynik(1), \dots, wynik(1000)$.

Komentarz do zadania

1.1.

Do rozwiązania tego zadania stosujemy definicję rekurencyjną $wynik(i)$, wynikającą wprost z podanego pseudokodu:

$$\begin{aligned} wynik(0) &= wynik(1) = wynik(2) = 1 \\ wynik(i) &= wynik(i-3) + wynik(i-1) + 1 \quad \text{dla parzystych } i > 2 \\ wynik(i) &= wynik(i-1) \bmod 7 \quad \text{dla nieparzystych } i > 2 \end{aligned}$$

A zatem:

- $wynik(3) = wynik(2) \bmod 7 = 1 \bmod 7 = 1$
- $wynik(4) = wynik(3) + wynik(1) + 1 = 1 + 1 + 1 = 3$

- $wynik(5) = wynik(4) \bmod 7 = 3 \bmod 7 = 3$
- $wynik(6) = wynik(5) + wynik(3) + 1 = 3 + 1 + 1 = 5$
- $wynik(7) = wynik(6) \bmod 7 = 5 \bmod 7 = 5$
- $wynik(8) = wynik(7) + wynik(5) + 1 = 5 + 3 + 1 = 9$

Zwróćmy uwagę, że w powyższym rozwiązaniu dla kolejnych argumentów (większych niż 2) funkcja *wynik* jest wywoływana wielokrotnie. Nie musimy jej jednak wielokrotnie wyznaczać, np. raz obliczony *wynik*(3) możemy wykorzystać przy każdym odwołaniu, bez ponownego obliczania.

1.2.

Najpierw rozwiążemy drugą część zadania, czyli podamy zależność rekurencyjną, określającą złożoność elementarną. Na podstawie tej zależności łatwo rozwiążemy pierwszą część zadania.

Ponieważ wykonania *wynik*(0), *wynik*(1) oraz *wynik*(2) kończą się natychmiastowym zwróceniem wyniku, każdemu z nich odpowiada dokładnie jedno wykonanie elementarne. To daje nam warunek brzegowy rekurencji:

$$E(0) = E(1) = E(2) = 1.$$

Natomiast dla każdego $i > 2$ sytuacja jest zależna od parzystości liczby i :

- Dla parzystych $i > 2$ wykonanie *wynik*(i) pociąga za sobą wykonanie *wynik*($i-1$) oraz *wynik*($i-3$). Zgodnie z przyjętymi oznaczeniami wykonanie *wynik*($i-1$) wymaga $E(i-1)$ wykonień elementarnych. Analogicznie *wynik*($i-3$) wymaga $E(i-3)$ wykonień elementarnych. Zatem *wynik*(i) pociąga za sobą $E(i-1) + E(i-3)$ wykonień elementarnych. Stąd:

$$E(i) = E(i-1) + E(i-3) \text{ dla parzystego } i > 2.$$

- Z kolei wykonanie *wynik*(i) dla nieparzystego $i > 2$ powoduje tylko wykonanie *wynik*($i-1$). Stąd:

$$E(i) = E(i-1) \text{ dla nieparzystego } i > 2.$$

Ostatecznie:

$$E(i) = \begin{cases} 1, & \text{dla } i \in \{0, 1, 2\}, \\ E(i-3) + E(i-1), & \text{dla parzystego } i > 2, \\ E(i-1), & \text{dla nieparzystego } i > 2, \end{cases}$$

Znając rekurencyjną formułę dla wyznaczania $E(i)$, możemy wypełnić podaną w zadaniu tabelkę, stosując tę formułę dla kolejnych $i=3, 4, \dots, 10$, podobnie jak w zadaniu 1 robiliśmy to dla funkcji *wynik*. Drobna różnica polega jedynie na tym, że w zadaniu 1 formuła zapisana była w innej postaci, a mianowicie w postaci pseudokodu.

1.3.

Naturalnym sposobem uniknięcia kolejnych wywołań rekurencyjnych jest tablicowanie wyników, co zostało zasugerowane w podanym niekompletnym algorytmie. Jeśli wartości *wynik*(0), *wynik*(1), ..., *wynik*($i-1$) przechowywane są w komórkach $W[0], W[1], \dots, W[i-1]$ tablicy W , to wartość *wynik*(i) możemy wyznaczyć jako $W[i-3] + W[i-1] + 1$ dla i parzystych oraz $W[i] \bmod 7$ dla i nieparzystych. Aby wyznaczoną wartość zapamiętać w odpowiedniej ko-

mórce tablicy W , użyjemy powyższych wyrażeń do uzupełnienia instrukcji podstawienia w podanym algorytmie.

Zmienna \max_wart , jak wskazuje nazwa, może być użyta do przechowywania największej wyznaczonej dotychczas wartości funkcji $wynik$, zatem aktualizujemy ją zawsze, gdy $W[i] > \max_wart$. Poniżej prezentujemy kompletne rozwiązanie:

```

 $W[0] \leftarrow 1$ 
 $W[1] \leftarrow 1$ 
 $W[2] \leftarrow 1$ 
 $\max\_wart \leftarrow 1$ 
dla  $i = 3, 4, \dots, 1\,000$  wykonuj
    jeżeli  $i \bmod 2 = 0$ 
         $W[i] \leftarrow W[i-1] + W[i-3] + 1$ 
    w przeciwnym razie
         $W[i] \leftarrow W[i] - W[i-1] \bmod 7$ 
    jeżeli  $W[i] > \max\_wart$ 
         $W[i] \leftarrow \max\_wart$ 
zwróć  $\max\_wart$ 
```

Rozwiązańe

1.1.

i	$wynik(i)$
2	1
3	1
4	3
5	3
6	5
7	5
8	9

1.2.

Poprawne wartości $E(i)$ dla argumentów podanych w tabeli

i	$E(i)$
0	1
3	1
5	2
7	3
9	5
10	8

Poprawna definicja rekurencyjna:

$$E(0) = E(1) = E(2) = 1$$

$$E(i) = E(i-1) + E(i-3) \text{ dla parzystego } i > 2$$

$$E(i) = E(i-1) \text{ dla nieparzystego } i > 2$$

1.3.

```
W[0] ← 1
W[1] ← 1
W[2] ← 1
max_wart ← 1
dla i = 3, 4, ..., 1000 wykonuj
    jeżeli i mod 2 = 0
        W[i] ← W[i - 1] + W[i - 3] + 1
    w przeciwnym razie
        W[i] ← W[i - 1] mod 7
    jeżeli W[i] > max_wart
        W[i] ← max_wart
zwróć max_wart
```

Zadanie 2.

Wiązka zadań Ułamki dwójkowe

W systemach pozycyjnych o podstawie innej niż 10 można zapisywać nie tylko liczby całkowite, ale również rzeczywiste z pewną dokładnością. Na przykład w systemie dwójkowym cyfry po przecinku odpowiadają kolejnym potęgom 1/2 (jednej drugiej). Cyfra 1 na pierwszym miejscu po przecinku odpowiada 1/2, na drugim miejscu — 1/4, na trzecim — 1/8 i tak dalej.

Na przykład $(0,101)_2 = 1/2 + 1/8 = 5/8 = 0,625_{10}$. Podobnie jak w systemie dziesiętnym nie każda liczba daje się zapisać w ten sposób dokładnie — na przykład liczba 1/3 nie ma skończonego rozwinięcia w systemie dwójkowym (ani też w dziesiętnym). Można jednak stosunkowo łatwo wyznaczyć zadaną liczbę początkowych cyfr po przecinku dla każdej liczby rzeczywistej.

Następujący algorytm przyjmuje na wejściu liczbę rzeczywistą x należącą do przedziału $[0, 1)$ oraz dodatnią liczbę całkowitą k i wypisuje k pierwszych cyfr liczby x w zapisie dwójkowym. Przeanalizuj algorytm i odpowiedz na podane pytania.

Dane:

x — liczba rzeczywista, $0 \leq x < 1$,
 k — liczba całkowita dodatnia.

Wynik:

zapis dwójkowy liczby x do k -tego miejsca po przecinku.

```
funkcja binarny(x, k)
    wypisz „0,”
    y ← x
    dla i=1, 2, ..., k wykonuj
        (*)     jeżeli y ≥ 1/2
                wypisz „1”
                w przeciwnym razie
                    wypisz „0”
                    y ← y * 2
        jeżeli y ≥ 1
            y ← y - 1
```

2.1.

Podaj liczbę wypisaną przez algorytm dla $x = 0.6$, $k = 5$ oraz wartości zmiennej y przy każdorazowym wykonaniu wiersza oznaczonego (*).

Kolejne wykonanie (*)	Wartość zmiennej y
1	
2	
3	
4	
5	

Liczba wypisana przez algorytm:

2.2.

Podaj przykład liczby x , dla której po wykonaniu funkcji $\text{binarny}(x, 4)$ zmienna y ma wartość 0, a po wykonaniu funkcji $\text{binarny}(x, 3)$ zmienna y nie jest równa 0.

2.3.

W systemie trójkowym używa się cyfr 0, 1 i 2. Cyfra 1 na pierwszym miejscu po kropce oznacza $1/3$, zaś 2 oznacza $2/3$. Na drugim miejscu są to odpowiednio $1/9$ i $2/9$, na trzecim — $1/27$ i $2/27$ i tak dalej, z kolejnymi potęgami trójki w mianownikach.

Poniżej podany jest algorytm wypisujący dla zadanej liczby rzeczywistej x z przedziału $[0,1)$ oraz liczby całkowitej dodatniej k pierwsze k cyfry zapisu x w systemie trójkowym. Uzupełnij luki tak, aby algorytm działał prawidłowo.

funkcja trójkowy(x , k)

```
wypisz „0,”  
y ← x  
dla i = 1, 2, ..., k wykonuj  
    jeżeli y ≥ 2/3  
        wypisz „2”  
    jeżeli .....  
        wypisz „1”  
    jeżeli .....  
        wypisz „0”  
    y = y * 3  
    jeżeli y ≥ 2  
        .....  
    jeżeli y ≥ 1  
        .....
```

Komentarz do zadania

2.1.

Algorytm zaczyna od wypisania zera i przecinka dziesiętnego. Następnie zaczyna się główna pętla: w pierwszej iteracji $y = 0.6$, a zatem pierwszą po przecinku cyfrą jest 1. Mnożymy y przez 2 i jeśli przekroczy 1, odejmujemy 1. Ponieważ $0.6 \cdot 2 = 1.2$, to po tej iteracji zmienna y

przybierze wartość 0,2. W kolejnym obrocie pętli wypiszemy cyfrę 0 (jako że $0,2 < 0,5$), po czym podwoimy y , otrzymując 0,4. Kontynuując w ten sposób działania, dojdziemy do odpowiedzi takiej jak wzorcowa.

Można przy okazji zauważyc, że po czwartej iteracji y znowu przybierze wartość 0,6, a zatem dalsze kroki algorytmu, gdybyśmy wykonali ich więcej, byłyby identyczne z pierwszymi czterema. Widać stąd, że $(0,6)_2 = 0,1001100110011\dots$

2.2.

Wartość zmiennej y równa zero oznacza po prostu, że już wszystkie dalsze cyfry dwójkowe liczby x są zerami, innymi słowy, że rozwinięcie dwójkowe x się skończyło.

W zadaniu pytamy zatem o liczbę, która po trzech iteracjach ma jeszcze dalsze niezerowe cyfry dwójkowe do wypisania, ale po czterech już nie ma. Musi to więc być liczba, która w rozwinięciu dwójkowym ma dokładnie cztery cyfry po przecinku. Najmniejszą taką liczbą jest $(0,0001)_2$, czyli $1/16 = 0,0625$. Innymi możliwymi odpowiedziami są na przykład $(0,0011)_2 = 3/16 = 0,1875$ albo $(0,1111)_2 = 15/16 = 0,9375$.

2.3.

Algorytm $\text{binarny}(x, k)$ opiera się na następującym pomyśle: jeśli liczba x jest nie mniejsza od $1/2$, to jej pierwszą cyfrą dwójkową po przecinku musi być 1. Jeśli teraz pomnożymy liczbę x przez 2, to odpowiada to przesunięciu całego rozwinięcia o jedno miejsce w lewo. Druga po przecinku cyfra liczby x to pierwsza cyfra po przecinku $2x$, czyli możemy ją wyznaczyć podobnie jak poprzednio: sprawdzając, czy jest większa lub równa $1/2$. Oczywiście musimy najpierw pominąć stojącą przed przecinkiem część całkowitą — odejmujemy zatem 1, jeśli liczba przekroczyła 1 w czasie mnożenia.

Bardzo podobną technikę stosujemy w algorytmie $\text{trojkowy}(x, k)$. Tutaj pierwsza cyfra po przecinku powinna być równa 2, jeśli liczba x jest nie mniejsza od $2/3$, 1 — jeśli x należy do przedziału $[1/3, 2/3)$, a 0 — jeśli x jest mniejsze od $1/3$. Następnie dokonujemy przesunięcia w lewo, mnożąc liczbę przez 3, po czym usuwamy z niej część całkowitą. Istotną różnicą jest to, że teraz część całkowita liczby może wynosić 0, 1 lub 2, zatem musimy odjąć 1 lub 2, zależnie od potrzeby:

```
dla i=1, 2, ..., k wykonuj
    jeżeli y ≥ 2/3
        wypisz „2”
    jeżeli y ≥ 1/3 oraz y < 2/3
        wypisz „1”
    jeżeli y < 1/3
        wypisz „0”
    y ← y * 3
    jeżeli y ≥ 2
        y ← y - 2
    jeżeli y ≥ 1
        y ← y - 1
```

Zamiast ostatnich czterech wierszy — odpowiadających pomijaniu części całkowitej — można było w oryginalnym algorytmie napisać tak:

```

jeżeli  $y \geq 2$ 
     $y \leftarrow y - 1$ 
jeżeli  $y \geq 1$ 
     $y \leftarrow y - 1$ 

```

Jeśli liczba będzie większa niż 2, algorytm najpierw odejmie 1, a następnie zauważ, że liczba wciąż jest większa niż 1, i odejmie znów jedynkę. Warto jeszcze wspomnieć, że gdyby luki nie narzucały konstrukcji algorytmu, można byłoby użyć krótszego zapisu:

```

dopóki  $y \geq 1$  wykonuj
     $y \leftarrow y - 1$ 

```

Miałoby to ten sam efekt: odrzucenie z liczby y jej części całkowitej.

Zadanie 3.

Wiązka zadań *Ciekawe mnożenie*

Dana jest następująca funkcja rekurencyjna:

Dane:

x — liczba całkowita,
 n — dodatnia liczba całkowita.

```

funkcja F(x, n)
    jeżeli n = 1
        podaj wynik x i zakończ
    w przeciwnym razie
        jeżeli n mod 3 = 0
            k ← F(x, n div 3)
            podaj wynik k*k*k i zakończ
        w przeciwnym razie
            podaj wynik x*F(x, n-1) i zakończ
(*)                                                 (**)

```

Uwaga: „div” jest operatorem dzielenia całkowitego.

3.1.

Podaj wszystkie wywołania rekurencyjne funkcji F oraz obliczany po każdym wywołaniu wynik, jeśli na początku wywołamy F(2, 10).

wywołanie	wynik
F(2, 10)	
F(;)	

3.2.

Uzupełnij tabelę o brakujące elementy:

x	n	wynik $F(x, n)$
2	2	4
2	3	
3		81
	5	32
2		256
	10	1024

3.3.

Uzupełnij tabelę, podając łączną liczbę mnożeń wykonanych w wierszach oznaczonych (*) i (**) po wywołaniu F dla podanych argumentów x i n :

x	n	Liczba operacji mnożenia
2	2	1
2	3	
3	4	
4	7	
4	8	
4	9	

3.4.

Podaj, która z poniższych funkcji określa liczbę wszystkich operacji mnożenia wykonywanych przez powyższy algorytm dla argumentu n będącego potągią trójką ($n = 3^m$ dla pewnego nieujemnego m):

- $lmnozen(n) = n \text{ div } 2$
- $lmnozen(n) = \log_2 n$
- $lmnozen(n) = 2 \cdot \log_3 n$
- $lmnozen(n) = 1 + \sqrt{n}$

Zadanie 4.

Wiązka zadań Silniowy system pozycyjny

Pojęcie *silni* dla liczb naturalnych większych od zera definiuje się następująco:

$$n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot (n - 1) \cdot n$$

Silniowy system pozycyjny to pozycyjny sposób zapisu liczb naturalnych, w którym mnożniki dla kolejnych pozycji są definiowane przez silnie kolejnych liczb naturalnych, tzn.

$$(x)_! = (x_n x_{n-1} x_{n-2} \dots x_2 x_1)_! = x_n \cdot n! + x_{n-1} \cdot (n - 1)! + \dots + x_2 \cdot 2! + x_1 \cdot 1!$$

W systemie silniowym współczynnik x_i , który odpowiada mnożnikowi $i!$, spełnia zależność $0 \leq x_i \leq i$.

Zapis każdej liczby w silniowym systemie pozycyjnym jest jednoznaczny, tzn. każdą liczbę naturalną można zapisać tylko w jeden sposób i każdą liczbę naturalną można zapisać dokładnie w jeden sposób.

Uwaga: W poniższych zadaniach będziemy mieć do czynienia tylko z takimi liczbami, dla których współczynniki x_i spełniają zależność $0 \leq x_i \leq 9$.

Przykład

$$(1220)_! = 1 \cdot 4! + 2 \cdot 3! + 2 \cdot 2! + 0 \cdot 1! = 24 + 12 + 4 + 0 = 40.$$

4.1.

Uzupełnij tabelę. Zamień zapis liczby w systemie silniowym na jej zapis w systemie dziesiętnym.

liczba w systemie silniowym	liczba w systemie dziesiętnym
$(310)_!$	
$(2011)_!$	
$(54211)_!$	

4.2.

Podaj zapis w systemie silniowym największej liczby, jaką można w tym systemie zapisać na pięciu pozycjach.

4.3.

Zamiana zapisu liczby w systemie dziesiętnym na zapis w systemie silniowym może przebiegać według następującego schematu: Szukamy największej liczby k , której silnia nie przekracza liczby x . Pierwsza jej cyfra to wynik dzielenia całkowitego x przez $k!$. Kolejne cyfry zapisu silniowego (zaczynając od cyfr najbardziej znaczących) otrzymujemy przez wyznaczanie wyników dzielenia liczby x przez $(k-1)!, (k-2)!, \dots, 2!, 1!$. Po wyznaczeniu cyfry x_i , odpowiadającej współczynnikowi $i!$, zmniejszamy wartość x o liczbę odpowiadającą cyfrze x_i , czyli $x_i \cdot i!$. Oznacza to, że x przyjmuje wartość $x \bmod k!$.

Przykład

x	k	$x \bmod k!$	$x \bmod k!$
1548	6	2	108
108	5	0	108
108	4	4	12
12	3	2	0
0	2	0	0
0	1	0	0

Liczba dziesiętna 1548 w zapisie silniowym: (204200)!

Wykonaj zamianę liczby 5489 z systemu dziesiętnego na silniowy zgodnie z opisany powyżej algorytmem. Uzupełnij poniższą tabelkę oraz podaj zapis silniowy liczby 5489.

x	k	$x \text{ div } k!$	$x \text{ mod } k!$
5489			

Liczba dziesiętna 5489 w zapisie silniowym:

4.4.

Poniżej przedstawiono algorytm z lukami, który zamienia zapis liczb z systemu dziesiętnego na system silniowy. Uzupełnij luki w tym algorytmie.

Specyfikacja

Dane:

x — liczba całkowita dodatnia zapisana w systemie dziesiętnym,

Wynik:

s — napis reprezentujący liczbę x zapisaną w systemie silniowym.

$silnia \leftarrow 1$

$k \leftarrow 1$

dopóki ($silnia < x$) **wykonuj**

$k \leftarrow k + 1$

$silnia \leftarrow silnia * k$

jeżeli

$silnia \leftarrow silnia \text{ div } k$

$k \leftarrow k - 1$

$s \leftarrow " "$

dopóki ($k > 0$) **wykonuj**

$cyfra \leftarrow$

$s \leftarrow s \circ \text{tekst} (cyfra)$

$x \leftarrow$

$silnia \leftarrow$

$k \leftarrow k - 1$

Uwaga

tekst (x) oznacza funkcję zamieniającą liczbę x na jej zapis tekstowy

” ” oznacza napis pusty

$u \circ v$ oznacza sklejenie dwóch napisów: u oraz v

Zadanie 5.

Wiązka zadań Sortowanie przez wstawianie na dwa sposoby

Sortowanie przez wstawianie polega na powtarzaniu operacji wstawiania elementu do już uporządkowanego ciągu. Aby znaleźć w uporządkowanym ciągu miejsce, w które należy wstawić nowy element, można stosować różne strategie. Poniższy algorytm znajduje to miejsce metodą wyszukiwania binarnego.

Specyfikacja

Dane:

n — liczba naturalna oznaczająca długość ciągu,

$A[1..n]$ — ciąg liczb całkowitych zapisanych w tablicy

Wynik:

$A[1..n]$ — tablica liczb całkowitych, w której liczby zostały ustawione w porządku niemalejącym

Algorytm

dla $j = n - 1, n - 2, \dots, 1$ **wykonuj**

$x \leftarrow A[j]$

$p \leftarrow j$

$k \leftarrow n + 1$

dopóki $k - p > 1$ **wykonuj**

$i \leftarrow (p + k) \text{ div } 2$

jeżeli $x \leq A[i]$

$k \leftarrow i$

w przeciwnym razie

$p \leftarrow i$

dla $i = j, j + 1, \dots, p - 1$ **wykonuj**

$A[i] \leftarrow A[i + 1]$

$A[p] \leftarrow x$

5.1.

Przeanalizuj działanie powyższego algorytmu dla ciągu 12, 4, 3, 10, 5, 11 o długości $n = 6$ i podaj, ile razy zostaną wykonane instrukcje $k \leftarrow i$ i $p \leftarrow i$ dla kolejnych wartości j zamieszczonych w tabeli.

Wartość j	Liczba wykonań	
	$k \leftarrow i$	$p \leftarrow i$
5		
4		
3		
2		
1		

5.2.

Uzupełnij luki w poniższym algorytmie sortowania przez wstawianie tak, aby znajdowanie miejsca na kolejny wstawiany element było realizowane metodą wyszukiwania liniowego.

Specyfikacja

Dane:

n — liczba naturalna oznaczająca długość ciągu, $A[1..n]$ — ciąg liczb całkowitych zapisanych w tablicy.

Wynik:

$A[1..n]$ — tablica liczb całkowitych, w której liczby zostały ustawione w porządku niemalejącym.

Algorytm:

```
dla  $j = n - 1, n - 2, \dots, 1$  wykonuj  
     $x \leftarrow \dots \dots \dots$   
     $i \leftarrow \dots \dots \dots$   
    dopóki ( $i \leq n$ ) i ( $x > A[i]$ ) wykonuj  
         $A[i - 1] \leftarrow A[i]$   
         $i \leftarrow i + 1$   
         $\dots \dots \dots \leftarrow x$ 
```

5.3.

Porównaj dwa algorytmy sortowania przez wstawianie: taki, w którym miejsce dla wstawianego elementu jest znajdowane metodą wyszukiwania binarnego, i taki, w którym jest ono znajdowane metodą wyszukiwania liniowego. Zaznacz znakiem X w odpowiedniej kolumnie, które zdanie jest prawdziwe (P), a które jest fałszywe (F).

Oba algorytmy dla ciągu 12, 4, 3, 10, 5, 11 wykonują

	P	F
jednakową liczbę porównań między elementami ciągu liczb.		
jednakową liczbę przesunięć elementów w tablicy.		
tylko samo powtórzenie pętli zewnętrznej w algorytmie.		
jednakową liczbę instrukcji podstawienia wartości do zmiennej x .		

Zadanie 6.

Wiązka zadań *Od szczegółu do ogółu*

Rozważmy następujący algorytm:

Dane:

k — liczba naturalna,
 $A[1\dots 2^k]$ — tablica liczb całkowitych.

Algorytm 1:

```

 $n \leftarrow 1$ 
dla  $i=1,2,\dots,k$  wykonuj
     $n \leftarrow 2 \cdot n$ 
     $s \leftarrow 1$ 
    dopóki  $s < n$  wykonuj
         $j \leftarrow 1$ 
        dopóki  $j < n$  wykonuj
            (*)
            (**)   jeżeli  $A[j] > A[j+s]$ 
                    zamień( $A[j], A[j+s]$ )
            (*)    $j \leftarrow j + 2 \cdot s$ 
            (***)    $s \leftarrow 2 \cdot s$ 
    zwróć  $A[1]$ 

```

Uwaga: Funkcja $\text{zamień}(A[j], A[j+s])$ zamienia miejscami wartości $A[j]$ oraz $A[j+s]$.

6.1.

Prześledź działanie algorytmu 1 dla podanych poniżej wartości k i początkowych zawartości tablicy A . W każdym wierszu poniższej tabeli wpisz końcową zawartość tablicy A .

k	Początkowa zawartość tablicy $A[1\dots 2^k]$	Końcowa zawartość tablicy $A[1\dots 2^k]$
2	[4, 3, 1, 2]	[1, 4, 3, 2]
2	[2, 3, 4, 1]	
3	[1, 2, 3, 4, 5, 6, 7, 8]	
3	[8, 7, 6, 5, 4, 3, 2, 1]	
3	[4, 5, 6, 1, 8, 3, 2, 4]	

6.2.

Wskaż, które z poniższych zdań są prawdziwe (P), a które fałszywe (F), wstawiając znak X w odpowiedniej kolumnie:

	P	F
Po zakończeniu działania algorytmu 1 komórka $A[2^k]$ zawiera największą z liczb $A[1], \dots, A[2^k]$.		
Po zakończeniu działania algorytmu 1 spełniona jest nierówność $A[i] \leq A[i+1]$ dla każdego i , takiego że $1 \leq i \leq 2^k$.		
Po zakończeniu działania algorytmu 1 komórka $A[1]$ zawiera najmniejszą z liczb $A[1], \dots, A[2^k]$.		

6.3.

Wskaż, które z poniższych zdań są prawdziwe (P), a które fałszywe (F), wstawiając znak X w odpowiedniej kolumnie. Przyjmij, że $n=2^k$ oraz $k>1$:

	P	F
Instrukcja jeżeli w wierszu (*) jest wykonywana mniej niż $2n$ razy.		
Instrukcja jeżeli w wierszu (*) jest wykonywana mniej niż $n/2$ razy.		
Możliwe jest dobranie takiej początkowej zawartości $A[1..2^k]$, że instrukcja zamiany z wiersza (**) nie zostanie wykonana ani razu.		
Możliwe jest dobranie takiej początkowej zawartości $A[1..2^k]$, że instrukcja zamiany z wiersza (**) zostanie wykonana co najmniej $2n^2$ razy.		

6.4.

Rozważmy poniższy algorytm podobny do **algorytmu 1**.

Wejście: k — liczba naturalna,
 $A[1..2^k]$ — tablica liczb całkowitych.

Algorytm 2:

```
 $n \leftarrow 1$ 
dla  $i=1,2,\dots,k$  wykonuj
     $n \leftarrow 2 \cdot n$ 
     $s \leftarrow 1$ 
    dopóki  $s < n$  wykonuj
         $j \leftarrow 1$ 
        dopóki  $j < n$  wykonuj
            (*)
                jeżeli  $A[j] > A[j+1]$ 
                    zamień( $A[j], A[j+1]$ )
                 $j \leftarrow j+1$ 
             $s \leftarrow s+1$ 
        zwróć  $A[1], A[2], \dots, A[n]$ 
```

Uwaga: Funkcja $\text{zamień}(A[j], A[j+1])$ zamienia miejscami wartości $A[j]$ oraz $A[j+1]$.

Uzupełnij luki w poniższych zdaniach. Przyjmij $n=2^k$ oraz $k>1$.

Po zakończeniu działania algorytmu 2 element $A[i]$ jest

niż element $A[i+1]$ dla każdego i większego od

oraz mniejszego od.....

Wiersz (*) **algorytmu 2** wykonywany będzie w przebiegu algorytmu

- niż n razy,
- niż n^2 razy.

Zadanie 7.

Wiązka zadań Scalanie

Rozważmy następujący algorytm, który jako dane przyjmuje tablicę n -elementową, gdzie n jest potęgą dwójki:

Dane: tablica liczb rzeczywistych $T[1..n]$, gdzie $n = 2^m$, a m jest liczbą całkowitą nieujemną

funkcja uporządkuj($T[1..n]$):

 jeżeli $n=1$

 zwróć $T[1..n]$ i zakończ

$k \leftarrow n/2$

$A[1..k] \leftarrow \text{uporządkuj}(T[1 .. k])$

$B[1..k] \leftarrow \text{uporządkuj}(T[k+1 .. n])$

 zwróć scal(A, B) i zakończ

Funkcja $\text{scal}(A, B)$ dla danych dwóch tablic o rozmiarze k zwraca tablicę o rozmiarze $2k$, powstałą przez połączenie tablic A i B w sposób uporządkowany, tj. od elementu najmniejszego do największego. Na przykład dla tablic $A = [4, 6, 18, 22]$ i $B = [1, 3, 10, 15]$ wywołanie $\text{scal}(A, B)$ zwróci tablicę $[1, 3, 4, 6, 10, 15, 18, 22]$.

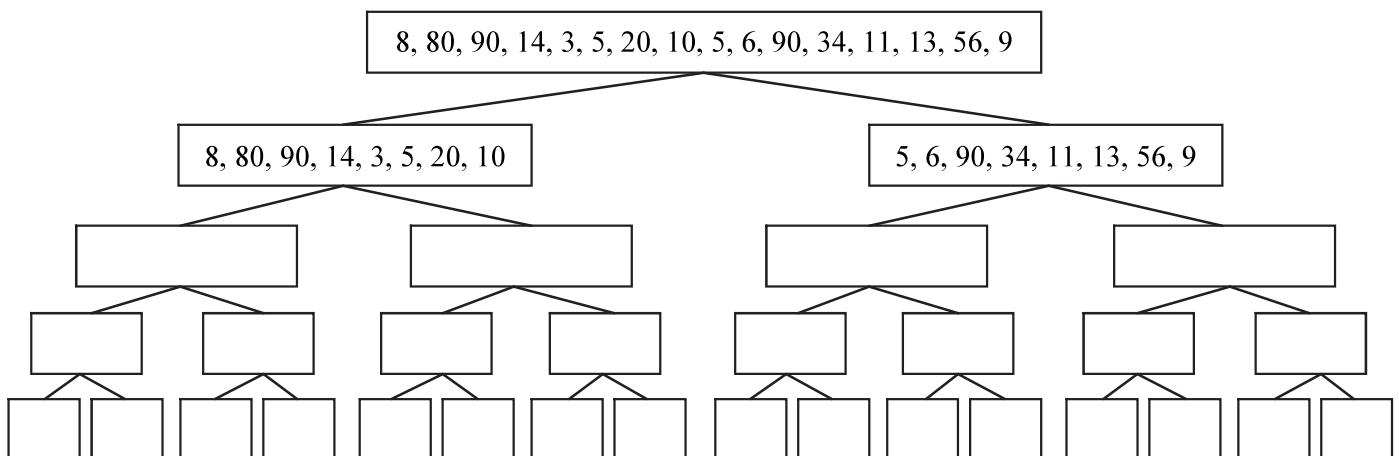
7.1.

Spośród danych tablic wybierz te, które są zgodne ze specyfikacją algorytmu, a następnie podaj dla nich wynik jego działania.

- $T = [15, 11]$,
- $T = [1, 3, 8]$,
- $T = [8, 4, 2, 1]$,
- $T = [10, 15, 1, 6, 9, 2, 5, 90]$.

7.2.

Funkcja uporządkuj jest funkcją rekurencyjną. Uzupełnij poniższe drzewo wywołań rekurencyjnych dla danej tablicy $T = [8, 80, 90, 14, 3, 5, 20, 10, 5, 6, 90, 34, 11, 13, 56, 9]$.



7.3.

Założymy, że wywołanie procedury $scal(A,B)$ dla dwóch tablic o długości k wykonuje $2k-1$ kosztownych operacji (porównywania liczb). Podaj liczbę kosztownych operacji, jaka zostanie wykonana przez funkcję *uporządkuj* dla tablicy

$$T[1..16] = [8, 80, 90, 14, 3, 5, 20, 10, 5, 6, 90, 34, 11, 13, 56, 9].$$

Zadanie 8.

Wiązka zadań *Dwa ciągi*

Niech $A[1..n]$ i $B[1..n]$ będą uporządkowanymi rosnąco tablicami liczb całkowitych i niech x będzie liczbą całkowitą. Rozważmy następujący algorytm:

$i \leftarrow 1$
 $j \leftarrow n$
dopóki ($i \leq n$ oraz $j > 0$) **wykonuj**
dopóki ($i \leq n$ oraz $j > 0$) **wykonuj**
(*) **jeżeli** $A[i] + B[j] = x$
 podaj wynik PRAWDA i zakończ algorytm
 w przeciwnym razie
 jeżeli $A[i] + B[j] < x$
 $i \leftarrow i + 1$
 w przeciwnym razie
 $j \leftarrow j - 1$
podaj wynik FAŁSZ

8.1.

Uzupełnij poniższą tabelę. Podaj wynik działania algorytmu oraz liczbę porównań wykonanych w wierszu oznaczonym (*).

Tablica A	Tablica B	x	Wynik działania algorytmu	Liczba porównań w kroku (*)
3, 5, 12, 17	8, 10, 13, 14	21		
4, 6, 8, 10	5, 7, 9, 11	13		

8.2.

Przeanalizuj działanie zaprezentowanego algorytmu i uzupełnij poniższą specyfikację.

Dane:

n — dodatnia liczba całkowita

$A[1..n], B[1..n]$ — n -elementowe tablice liczb całkowitych, posortowane rosnąco

x — liczba całkowita

Wynik:

PRAWDA, gdy

FAŁSZ, gdy

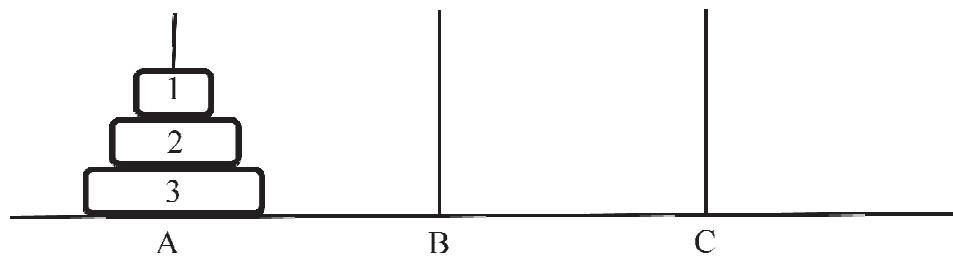
8.3.

Podaj przykład pięcioelementowych tablic A i B , dla których przy $x = 20$ algorytm wykona sześć porównań w wierszu (*), a wynikiem będzie *PRAWDA*.

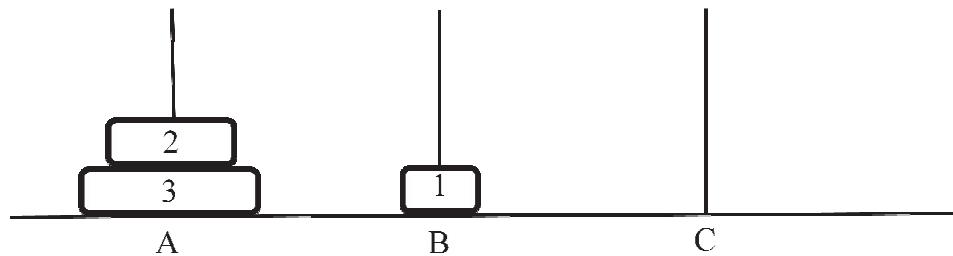
Zadanie 9.

Wiązka zadań *Wieże z Hanoi*

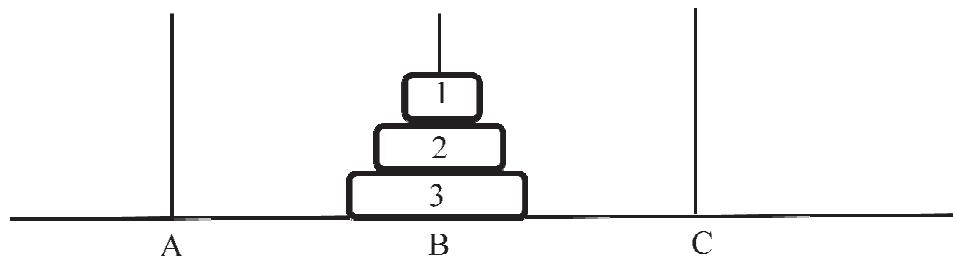
W problemie wież z Hanoi mamy trzy pręty oznaczone A, B i C oraz n okrągłych krążków o średnicach odpowiednio 1, 2, ..., n . Na początku wszystkie krążki nałożone są na pręt A, w kolejności od największego do najmniejszego (największy na dole, najmniejszy na górze). Układ ten (dla $n=3$) został przedstawiony na poniższym rysunku.



Zgodnie z regułami problemu krążki można przekładać między prętami. W jednym ruchu możliwe jest przełożenie krążka znajdującego się na szczycie jednego z prętów na szczyt innego pręta, pod warunkiem że nie kładziemy przekładanego krążka na krążek mniejszy od niego. Na przykład na poniższym rysunku krążek 2 możemy przełożyć z pręta A na pręt C, natomiast niemożliwe jest przełożenie go na pręt B.



Zadanie polega na przełożeniu wszystkich krążków z pręta A na pręt B, przy czym można korzystać z pomocniczego pręta C. Na poniższym rysunku przedstawiono efekt końcowy.



Problem wież z Hanoi można rozwiązać za pomocą algorytmu rekurencyjnego. W algorytmie pręty: startowy, docelowy i pomocniczy, podane są jako parametry wejściowe, odpowiednio x , y i z . Algorytm polega na tym, że najpierw przenosimy $n - 1$ krążków na pręt pomocniczy z , potem największy krążek zostaje przeniesiony na pręt docelowy y , a na koniec $n - 1$ krążków zostaje przeniesionych z pręta pomocniczego z na pręt docelowy y , przy czym pręt startowy x traktowany jest jako pomocniczy.

Algorytm

Specyfikacja

Dane:

- n — liczba całkowita dodatnia,
- x — nazwa pręta startowego,
- y — nazwa pręta docelowego,
- z — nazwa pręta pomocniczego.

Wynik:

ciąg ruchów opisujący rozwiązywanie problemu wież z Hanoi z n krążkami, w którym na początku wszystkie krążki znajdują się na pręcie x , a na końcu mają znaleźć się na pręcie y , zaś pomocniczym prętem jest z .

Uwaga: Pojedynczy ruch zapisujemy za pomocą znaku \Rightarrow . Na przykład $C \Rightarrow B$ oznacza przeniesienie krążka z pręta C na pręt B .

funkcja wieże(n , x , y , z)

jeżeli $n=1$

wypisz $x \Rightarrow y$

w przeciwnym razie

 wieże($n - 1$, x , z , y)

wypisz $x \Rightarrow y$

 wieże($n - 1$, z , y , x)

Przykład

Wywołanie $wieże(2, A, B, C)$ spowoduje dwa wywołania rekurencyjne: $wieże(1, A, C, B)$ oraz $wieże(1, C, B, A)$. Ciąg ruchów utworzony przez $wieże(2, A, B, C)$ ma postać:

$$A \Rightarrow C, A \Rightarrow B, C \Rightarrow B,$$

gdzie podkreślone ruchy są utworzone przez rekurencyjne wywołania $wieże(1, A, C, B)$ oraz $wieże(1, C, B, A)$.

9.1.

Podaj wszystkie wywołania rekurencyjne funkcji $wieże$ (wraz z ich parametrami), do których dojdzie w wyniku wywołania $wieże(3, A, B, C)$. Odpowiedź podaj w poniższej tabeli, uzupełniając parametry wszystkich wywołań rekurencyjnych.

n	x	y	z
3	A	B	C
2	A	C	B
1	A	B	C
1			
2			
1			
1			

9.2.

Prześledź działanie *wieże(3, A, B, C)* i uzupełnij poniżej wygenerowany ciąg ruchów:

$A \Rightarrow B$; $A \Rightarrow C$;

9.3.

Niech $H(n)$ oznacza liczbę ruchów wykonanych przez podany algorytm dla n krążków. Za- uważ, że rozwiązywanie problemu dla $n > 1$ krążków wymaga jednego ruchu oraz dwukrotnego rozwiązywania problemu dla $n - 1$ krążków. W oparciu o tę obserwację uzupełnij poniższą tabelę.

n	$H(n)$
1	1
2	3
3	
4	
5	
7	
10	

Podaj ogólny wzór określający liczbę ruchów dla n krążków:

$H(n) = \dots$

9.4.

Poniżej prezentujemy nierekurencyjne rozwiązywanie problemu wież z Hanoi.

Specyfikacja

Dane:

n — liczba całkowita dodatnia,

Wynik:

ciąg ruchów opisujący rozwiązywanie problemu wież z Hanoi z n krążkami, w którym na początku wszystkie krążki znajdują się na precie A, a na końcu powinny się znaleźć na precie B.

Algorytm:

- (1) **dopóki** (preć A jest niepusty lub preć C jest niepusty) **wykonuj**
- (2) **jeżeli** n jest parzyste:
 - (3) **przenieś** krążek nr 1 o jedną pozycję w lewo
 - (4) **w przeciwnym razie**
 - (5) **przenieś** krążek nr 1 o jedną pozycję w prawo
 - (6) **przenieś** krążek między prećmi, na których nie ma krążka nr 1

W powyższym algorytmie przeniesienie krążka nr 1 o jedną pozycję w prawo oznacza wykonanie jednego z ruchów $A \Rightarrow B$, $B \Rightarrow C$ lub $C \Rightarrow A$, tak aby krążek nr 1 został przeniesiony na inny preć. Analogicznie przeniesienie krążka w lewo oznacza wybór jednego z ruchów $A \Rightarrow C$, $B \Rightarrow A$ lub $C \Rightarrow B$, tak aby krążek nr 1 został przeniesiony na inny preć.

Ruch w kroku (6) powyższego algorytmu jest określony jednoznacznie, gdyż dopuszczalne jest tylko położenie mniejszego krążka na większym, a nie odwrotnie.

Przykład

Dla $n=3$ powyższy algorytm wykona następujący ciąg ruchów:

$$\underline{A \Rightarrow B}; A \Rightarrow C; \underline{B \Rightarrow C}; A \Rightarrow B; \underline{C \Rightarrow A}; C \Rightarrow B; \underline{A \Rightarrow B},$$

gdzie ruchy podkreślone przenoszą krążek nr 1 o jedną pozycję w prawo.

Wypisz ciąg ruchów, który poda powyższy algorytm dla $n=4$.

Zadanie 10.

Wiązka zadań *Dzielenie wielomianu*

Rozważamy wielomiany $P(x)$ stopnia $n - 1$, gdzie n jest potęgą dwójki:

$$P(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}, \quad n = 2^m.$$

Do obliczania wartości $P(x)$ można zastosować technikę „dziel i zwyciężaj” w następujący sposób: dzielimy postać ogólną wielomianu $P(x)$ na dwie części:

$$P(x) = \underbrace{a_0 + a_1x + \cdots + a_{k-1}x^{k-1}}_{\text{cz. I}} + \underbrace{a_kx^k + a_{k+1}x^{k+1} + \cdots + a_{n-1}x^{n-1}}_{\text{cz. II}},$$

gdzie $k = n/2$. Jeśli w drugiej części wydzielimy czynnik x^k , to otrzymamy równość

$$(*) \quad P(x) = A(x) + B(x) \cdot x^k,$$

gdzie $A(x)$ i $B(x)$ są wielomianami stopnia $k - 1$:

$$\begin{aligned} A(x) &= a_{k-1}x^{k-1} + \cdots + a_2x^2 + a_1x + a_0, \\ B(x) &= a_{n-1}x^{k-1} + \cdots + a_{k+2}x^2 + a_{k+1}x + a_k, \end{aligned}$$

W ten sposób problem obliczenia wartości $P(x)$ dzielimy na dwa podproblemy — obliczenia $A(x)$ i obliczenia $B(x)$, przy czym każdy z nich ma rozmiar o połowę mniejszy. Na przykład aby obliczyć wartość wielomianu ($n = 8$)

$$P(x) = -8 + 7x + 6x^2 - x^3 + 4x^4 + 5x^5 - 2x^6 + 3x^7,$$

obliczamy $k = n/2 = 4$, $A(x) = -8 + 7x + 6x^2 - x^3$ oraz $B(x) = 4 + 5x - 2x^2 + 3x^3$. Następnie korzystamy ze wzoru $P(x) = A(x) + B(x) \cdot x^4$.

Ponieważ do pełnej realizacji algorytmu we wzorze (*) potrzebne jest obliczenie wartości x^k , więc najpierw obliczamy wszystkie potęgi $x, x^2, x^4, \dots, x^{2^{m-1}}$ i zapamiętujemy je w tablicy $Z[0..m-1]$, np. za pomocą poniższej procedury.

Obliczanie tablicy $Z[0..m-1]$:

Dane:

n — liczba całkowita postaci 2^m , gdzie m jest liczbą całkowitą nieujemną,

x — liczba rzeczywista.

Wynik:

tablica $Z[0..m-1]$, dla której $Z[j] = x^{2^j}$

```

 $Z[0] \leftarrow x$ 
 $m \leftarrow 1$ 
 $w \leftarrow 2$ 
dopóki  $w < n$  wykonuj
     $Z[m] \leftarrow Z[m - 1] \cdot Z[m - 1]$ 
     $m \leftarrow m + 1$ 
     $w \leftarrow 2 \cdot w$ 

```

Mając tablicę Z , obliczamy wartość $P(x)$ za pomocą funkcji rekurencyjnej F .

Dane:

tablica liczb rzeczywistych $T[0..n-1]$ ze współczynnikami a_0, a_1, \dots, a_{n-1} , gdzie $n = 2^m$, a m jest liczbą całkowitą nieujemną;

tablica $Z[0..m-1]$, dla której $Z[j] = x^{2^j}$.

Wynik:

wartość $a_{n-1}x^{n-1} + \dots + a_2x^2 + a_1x + a_0$.

funkcja $F(T[0..n-1])$:

jeżeli $n=1$

zwróć $T[0]$ i zakończ

$k \leftarrow n/2$

$A[0..k-1] \leftarrow T[0..k-1]$

$B[0..k-1] \leftarrow T[k..n-1]$

zwróć $F(A) + F(B) \cdot Z[m-1]$ i zakończ

Prześledźmy na przykład obliczenie wartości wielomianu

$$P(x) = -8 + 7x + 6x^2 - x^3 + 4x^4 + 5x^5 - 2x^6 + 3x^7 \quad (n = 2^m, \quad m = 3)$$

dla $x=3$.

W pierwszym kroku algorytm obliczy tablicę $Z[0..2]$:

$$Z[0] = 3, \quad Z[1] = 3^2, \quad Z[2] = 3^4 = 81.$$

Następnie algorytm obliczy $F([-8,7,6,-1,4,5,-2,3])$, tzn.

$$F([-8,7,6,-1,4,5,-2,3]) = F([-8,7,6,-1]) + F([4,5,-2,3]) \cdot Z[2]$$

Obliczanie $F([4,5,-2,3])$ i $F([-8,7,6,-1])$ odbywa się w analogiczny sposób, tzn.

$$F([-8,7,6,-1]) = F([-8,7]) + F([6,-1]) \cdot Z[1]$$

$$\begin{aligned} &= F([-8]) + F([7]) \cdot Z[0] + (F([6]) + F([-1]) \cdot Z[0]) \cdot Z[1] \\ &= (-8) + 7 \cdot Z[0] + (6 + (-1) \cdot Z[0]) \cdot Z[1] \\ &= (-8) + 7 \cdot 3 + (6 + (-1) \cdot 3) \cdot 9 = 40, \end{aligned}$$

$$F([4,5,-2,3]) = F([4,5]) + F([-2,3]) \cdot Z[1]$$

$$\begin{aligned} &= F([4]) + F([5]) \cdot Z[0] + (F([-2]) + F([3]) \cdot Z[0]) \cdot Z[1] \\ &= 4 + 5 \cdot Z[0] + ((-2) + 3 \cdot Z[0]) \cdot Z[1] \\ &= 4 + 5 \cdot 3 + ((-2) + 3 \cdot 3) \cdot 9 = 82. \end{aligned}$$

Zatem

$$F([-8,7,6,-1,4,5,-2,3]) = F([-8,7,6,-1]) + F([4,5,-2,3]) * Z[2] = 40 + 82 * 81 = 6682.$$

Można zauważyć, że podczas obliczania $F([-8,7,6,-1,4,5,-2,3])$ łącznie zostanie wykonanych 7 mnożeń:

- 3 mnożenia podczas obliczania $F([-8,7,6,-1])=40$,
- 3 mnożenia podczas obliczania $F([4,5,-2,3])=82$,
- 1 mnożenie przy obliczaniu $F([-8,7,6,-1]) + F([4,5,-2,3]) * Z[2] = 40 + 82 * Z[2]$.

Liczba wszystkich wywołań rekurencyjnych jest równa 15. Oto wszystkie wywołania funkcji F :

- $F([-8,7,6,-1,4,5,-2,3])$
 - $F([-8,7,6,-1])$
 - $F([-8,7])$
 - $F([-8])$
 - $F([7])$
 - $F([6,-1])$
 - $F([6])$
 - $F([-1])$
 - $F([4,5,-2,3])$
 - $F([4,5])$
 - $F([4])$
 - $F([5])$
 - $F([-2,3])$
 - $F([-2])$
 - $F([3])$

10.1.

Podaj, jakie wartości zostaną obliczone w tablicy $Z[0 .. m-1]$ przy obliczaniu $P(2)$, gdzie

$$P(x) = 9 + x - 6x^3 + 2x^7 - 3x^{14} + 5x^{15}.$$

10.2.

Oblicz łączną liczbę operacji mnożenia, jaka zostanie wykonana przez funkcję $F(T)$ dla n -elementowej tablicy T.

Uzupełnij poniższą tabelkę:

n	Liczba operacji mnożenia
1	0
2	
4	
8	
16	
1024	

Uzupełnij poniższy wzór rekurencyjny dla $f(n)$ — liczby operacji mnożenia wykonywanych przez funkcję F dla tablicy n -elementowej:

$$f(n) = \dots \cdot f(n/2) + \dots \quad \text{dla } n > 1$$

10.3.

Wypisz wszystkie wywołania funkcji F podczas obliczania

$$P(x) = 9 + x - 6x^3 + 10x^5 + 2x^7 + 4x^9 - x^{10} - 3x^{14} + 5x^{15}.$$

Podaj wzór ogólny na $g(n)$ — liczbę wszystkich wywołań funkcji F dla wielomianu $P(x)$ stopnia $n - 1$, gdzie n jest potęgą dwójki. Uzupełnij poniższy wzór:

$$g(n) = \dots$$

10.4.

Do obliczania wartości wielomianu $P(x)$

$$P(x) = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1} \quad (n = 3^m),$$

a więc gdy n jest potągą trójką, można zastosować podział na trzy części

$$P(x) = \underbrace{a_0 + \dots + a_{k-1}x^{k-1}}_{\text{cz. I}} + \underbrace{a_kx^k + \dots + a_{2k-1}x^{2k-1}}_{\text{cz. II}} + \underbrace{a_{2k}x^{2k} + \dots + a_{n-1}x^{n-1}}_{\text{cz. III}},$$

gdzie $k = n/3$. Wówczas uzyskujemy

$$P(x) = A(x) + B(x) \cdot x^k + C(x) \cdot x^{2k} = A(x) + (B(x) + C(x) \cdot x^k) \cdot x^k,$$

gdzie $A(x), B(x), C(x)$ są wielomianami, w których liczba współczynników jest trzykrotnie mniejsza niż w wyjściowym wielomianie $P(x)$.

Wzorując się na funkcji F , możemy skonstruować rekurencyjną funkcję $G(T[0 .. n-1])$, obliczającą wartość $P(x)$ przez podział tablicy T na trzy równe części, o ile $n > 1$. Po obliczeniu trzech wyników dla każdej z części, powiedzmy $A(x), B(x), C(x)$, funkcja oblicza swój wynik, wykonując dodatkowo dwa mnożenia:

- jedno mnożenie przy obliczaniu $C(x) \cdot x^k$.
- jedno mnożenie przy obliczaniu $(B(x) + C(x) \cdot x^k) \cdot x^k$.

Podobnie jak poprzednio pomijamy problem obliczania potęg x^k dla $k = 3^j$ ($j \geq 0$), tzn. przyjmujemy, że potęgi te są przechowywane w pewnej tablicy $Z[0 .. m-1]$, a więc ich obliczanie nie jest wliczane do kosztu obliczeniowego funkcji G .

W ten sposób na przykład dla $n=3$ funkcja G wykona dokładnie 2 mnożenia, a dla $n=9$ funkcja wykona łącznie 8 mnożeń:

- po 2 mnożenia dla każdej z trzech części $A(x), B(x), C(x)$,
- jedno mnożenie przy obliczaniu $C(x) \cdot x^k$.
- jedno mnożenie przy obliczaniu $(B(x) + C(x) \cdot x^k) \cdot x^k$.

Uzupełnij poniższą tabelkę, podając liczbę mnożeń, jaka zostanie wykonana przez funkcję G dla n -elementowej tablicy współczynników T .

n	Liczba operacji mnożenia
3	2
9	8
27	
81	
243	

Zadanie 11.

Wiązka zadań *Odwrotna notacja polska*

Dla przetwarzania przez komputer wygodnym sposobem zapisu wyrażeń arytmetycznych jest tzw. odwrotna notacja polska (ONP). Zapis w ONP wyrażenia W nazywamy *postacią ONP* i oznaczamy ją $\text{ONP}(W)$. W ONP operator (dodawania, odejmowania, mnożenia, dzielenia) umieszczamy za jego argumentami, np. $2+5$ zapisujemy jako $2\ 5\ +$. Dokładniej, postać ONP dla wyrażenia definiujemy rekurencyjnie w następujący sposób:

1. Jeżeli W jest liczbą, to jego postać ONP jest równa W .
2. Jeżeli W ma postać $W_1 \ op \ W_2$, gdzie op jest operatorem, a W_1 i W_2 wyrażeniami, to $\text{ONP}(W)$ jest równe $\text{ONP}(W_1) \ \text{ONP}(W_2) \ op$.

Przykład

$W = W_1 \ op \ W_2$	W_1	W_2	op	$\text{ONP}(W)$
$1 + 2$	1	2	+	1 2 +
$5 - 7$	5	7	-	5 7 -
$3 * (5 - 7)$	3	$5 - 7$	*	3 5 7 - *
$(1 + 2) + (3 * (5 - 7))$	$1 + 2$	$3 * (5 - 7)$	+	1 2 + 3 5 7 - * +

Zauważmy, że dla $W = (1 + 2) + (3 * (5 - 7))$ wartość $\text{ONP}(W)$ uzyskujemy z połączenia $\text{ONP}(1 + 2) = 1 2 +$, $\text{ONP}(3 * (5 - 7)) = 3 5 7 - *$ oraz znaku dodawania $+$.

11.1.

Uzupełnij poniższą tabelę, podając dla każdego wyrażenia z pierwszej kolumny jego podwyrażenia, łączący je operator oraz postać ONP tego wyrażenia.

$W = W_1 \ x \ W_2$	W_1	W_2	op	$\text{ONP}(W)$
$4 + 3$	4	3	+	4 3 +
$(4 + 3) * 2$				
$5 * (7 - 6)$				
$((4 + 3) * 2) - (5 * (7 - 6))$				

11.2.

Postać ONP wyrażeń, choć dla ludzi mało czytelna, ma własności bardzo przydatne dla analizy komputerowej. W ONP nie są potrzebne nawiasy, a do wyznaczania wartości wyrażenia można zastosować prosty algorytm podany poniżej.

Specyfikacja

Dane:

n — liczba całkowita dodatnia,

$X = X[1 \dots n]$ — wyrażenie w ONP, gdzie $X[i]$ dla $1 \leq i \leq n$ jest liczbą lub znakiem ze zbioru $\{+, -, *\}$.

Wynik:

wartość wyrażenia X .

Algorytm:

$k \leftarrow 1$

dla $i=1,2,\dots,n$ **w wykonuj**

jeżeli $X[i]$ jest liczbą

$T[k] \leftarrow X[i]$

jeżeli $X[i] \in \{+, -, *\}$

$b \leftarrow T[k-1]$

$a \leftarrow T[k-2]$

$k \leftarrow k-2$

jeżeli $X[i] = '+'$

$T[k] \leftarrow a + b$

jeżeli $X[i] = '-'$

$T[k] \leftarrow a - b$

jeżeli $X[i] = '*'$

$T[k] \leftarrow a * b$

$k \leftarrow k+1$

zwróć $T[1]$

Prześledź działanie podanego algorytmu dla wyrażenia $X = 9\ 7\ +\ 3\ *\ 5\ 4\ -\ 2\ *\ -$, czyli dla $n=11$ oraz następujących wartości $X[1], \dots, X[11]$:

i	1	2	3	4	5	6	7	8	9	10	11
$X[i]$	9	7	+	3	*	5	4	-	2	*	-

Uzupełnij poniższą tabelę:

i	Wartość zmiennej k po i -tym przebiegu pętli	Zawartość tablicy $T[1..k-1]$ po i -tym przebiegu pętli
1	2	9
2	3	9, 7
4	3	16, 3
5		
6		
10		
11		

11.3.

Poniższy algorytm sprawdza, czy podany na wejściu ciąg liczb i operatorów jest poprawnym wyrażeniem w ONP.

Specyfikacja

Dane:

n — liczba całkowita dodatnia

$X = X[1..n]$ — ciąg elementów, z których każdy jest liczbą lub znakiem ze zbioru $\{+, -, *\}$.

Wynik:

Tak — jeśli X jest poprawnym wyrażeniem w ONP,

Nie — w przeciwnym przypadku.

Algorytm:

$licznik \leftarrow 0$

dla $i=1,2,\dots,n$ **w wykonuj**

jeżeli $X[i]$ jest liczbą

$licznik \leftarrow licznik + 1$

jeżeli $X[i] \in \{+, -, *\}$

jeżeli $licznik < 2$

 zwróć „Nie” i zakończ działanie

w przeciwnym razie

$licznik \leftarrow licznik - 1$

jeżeli $licznik \neq 1$

 zwróć „Nie”

w przeciwnym razie

 zwróć „Tak”

Oceń, które z podanych poniżej napisów są wyrażeniami zapisanymi poprawnie w ONP, wpisując słowa TAK lub NIE w trzeciej kolumnie poniższej tabeli. W drugiej kolumnie podaj wartości zmiennej $licznik$ po zakończeniu działania algorytmu dla poszczególnych napisów.

Napis	Wartość zmiennej $licznik$ po zakończeniu algorytmu	Czy poprawne wyrażenie w ONP?
1 2 + *	1	NIE
1 2 + 3 4 - 5 * 7 8 + 9		
1 2 3 4 5 + + + +		
1 2 3 4 5 + + + + +		
1 2 3 4 5 + + + + +		
1 2 + 2 3 - 3 4 * 4 5 + - - -		
1 2 + 2 3 - 3 4 * 4 5 + - - - -		
1 2 + 3 4 - 5 * 7 8 + 9 + + +		

11.4.

W poniższych wyrażeniach przyjmujemy, że op_1, \dots, op_{10} to znaki ze zbioru $\{+, -, *\}$. Podaj postać ONP poniższych wyrażeń.

X: (((((((1 op₁ 2) op₂ 3) op₃ 4) op₄ 5) op₅ 6) op₆ 7) op₇ 8) op₈ 9) op₉ 10)

ONP(X):

Y: (1 op₁ (2 op₂ (3 op₃ (4 op₄ (5 op₅ (6 op₆ (7 op₇ (8 op₈ (9 op₉ 10))))))))

ONP(Y):

Zadanie 12.

Wiązka zadań *Szyfr Beauforta*

Każdej z wielkich liter angielskiego alfabetu przyporządkowany jest kod ASCII, będący ośmiobitową liczbą naturalną. Poniższa tabela przedstawia kody ASCII dla liter od A do Z:

65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Dany jest następujący algorytm szyfrujący:

Dane:

n — liczba naturalna, wielkość tablicy *Tekst[]*

Tekst[] — tablica zawierająca kody ASCII liter tekstu do zaszyfrowania, elementy tablicy są numerowane od 0 do $n-1$

k — nieujemna liczba całkowita

Wynik:

Szyfrogram[] — tablica o długości n zawierająca kody ASCII liter zaszyfrowanego tekstu

```
i ← 0
dopóki (i < n) wykonuj
    (*)
        Szyfrogram[i] ← (90 - Tekst[i] + k) mod 26 + 65
        i ← i + 1
        k ← k + i
```

12.1.

Uzupełnij poniższą tabelę wartościami zmiennych i oraz k w wierszu oznaczonym (*) w kolejnych iteracjach algorytmu dla n równego 7:

Lp	i	k
1	0	20
2		
3		
4		
5		
6		

12.2.

Ile dla przedstawionego algorytmu jest możliwych początkowych wartości k , dających w efekcie różne szyfrogramy? Odpowiedź krótko uzasadnij.

12.3.

Uzupełnij poniższą tabelę. W wierszu pierwszym zaszyfruj przedstawionym w treści zadania algorytmem podany tekst jawnny, w wierszu drugim odszyfruj szyfrogram zakodowany tym algorytmem.

k	<i>Tekst[]</i>	<i>Szyfrogram[]</i>
1	MAPA	
14		DAN

12.4.

Używając pseudokodu lub wybranego języka programowania, napisz algorytm dekodujący szyfrogram.

Dane:

n — liczba naturalna, wielkość tablicy *Tekst[]*

Szyfrogram[] — tablica o długości n zawierająca kody ASCII liter zaszyfrowanego tekstu

k — nieujemna liczba całkowita

Wynik:

Tekst[] — tablica zawierająca kody ASCII liter odszyfrowanego tekstu, elementy tablicy są numerowane od 0 do $n-1$

Zadanie 13.

Wiązka zadań Zbiór punktów

Na płaszczyźnie kartezjańskiej danych jest n punktów: P_1, P_2, \dots, P_n oraz jeszcze jeden punkt A.

Współrzędne punktów zapisane są w tablicach $PX[1..n], PY[1..n]$ — współrzędne punktu P_i dane są jako para $(PX[i], PY[i])$ dla $i = 1, 2, \dots, n$. Współrzędne punktu A to (ax, ay) . Przeanalizuj poniższy algorytm i wykonaj poniższe polecenia.

Dane:

$PX[1..n], PY[1..n]$ — tablice liczb całkowitych zawierające współrzędne kolejnych punktów P_1, P_2, \dots, P_n

ax, ay — współrzędne punktu A.

Wynik:

odpowiedź TAK lub NIE.

funkcja sprawdz(PX[1..n], PY[1..n], ax, ay)

dla i = 1, 2, ..., n wykonuj

 roznicax \leftarrow PX[i] – ax

 roznicay \leftarrow PY[i] – ay

 drugix \leftarrow ax – roznicax

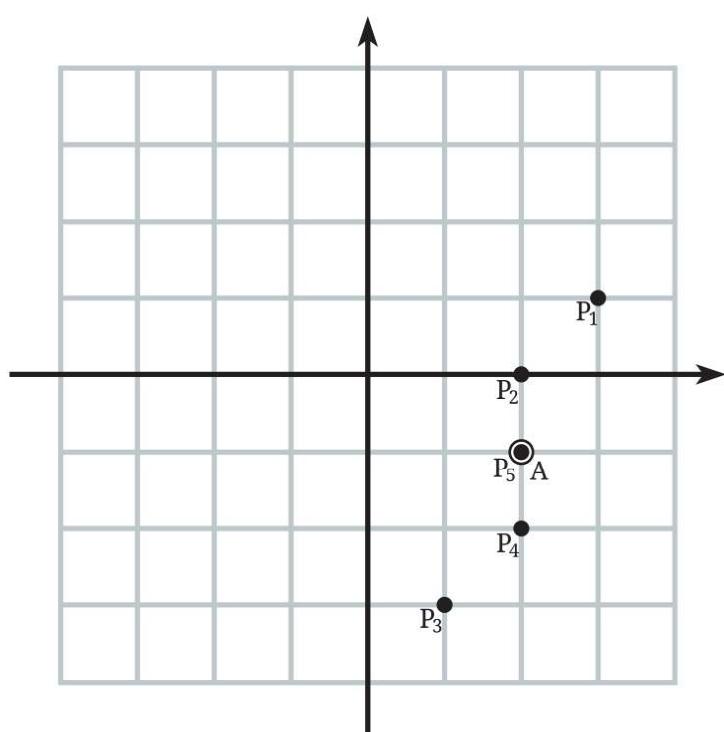
drugiy \leftarrow ay – roznicay
 znalezione \leftarrow fałsz
dla k = 1, 2, ..., n **wykonuj**
jeżeli drugix = PX[k] **oraz** drugiy = PY[k]
 znalezione \leftarrow prawda

jeżeli znalezione = fałsz
wypisz NIE i zakończ wykonywanie algorytmu
wypisz TAK

13.1.

Dla poniższych zbiorów punktów znajdź odpowiedź, jaką otrzymamy po wykonaniu powyższego algorytmu:

	P ₁ (-2,3) P ₂ (0,3) P ₃ (-2,2) P ₄ (0,2) P ₅ (-2,-1) P ₆ (0,-1) A (-1,1)	<i>Odpowiedź algorytmu:</i>



P₁ (3,1)
 P₂ (2,0)
 P₃ (1,-3)
 P₄ (2,-2)
 P₅ (2,-1)
 A (2,-1)

Odpowiedź algorytmu:

13.2.

Dane są punkty:

P₁ (3,-1)
 P₂ (0,2)
 P₃ (1,4)
 P₄ (1,3)
 P₅ (4,0)

oraz punkty P₆ i A o nieznanych współrzędnych. Wiadomo, że algorytm odpowiada TAK dla n = 6 oraz punktów P₁,..., P₆ i A. Podaj współrzędne punktów P₆ i A.

Zadanie 14.

Wiązka zadań Działka

Pan G jest geodetą. Jego zadaniem jest wyznaczenie pola powierzchni działki budowlanej o kształcie pewnego wielokąta wypukłego. Aby wyznaczyć pole powierzchni działki, pan G obchodzi ją dookoła, tak aby działkę mieć zawsze po swojej prawej stronie. Podczas pomiaru zapisuje współrzędne kolejnych wierzchołków wielokąta, przy czym:

1. rozpoczyna obchód w punkcie $P_1 = [x_1, y_1]$;
2. współrzędne kolejno odwiedzonych wierzchołków oznacza:

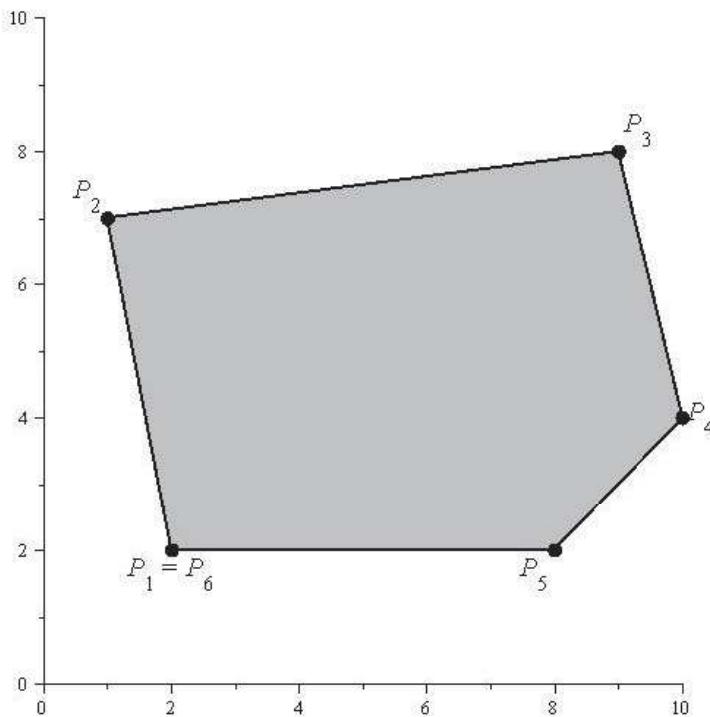
$$P_2 = [x_2, y_2], P_3 = [x_3, y_3], \dots, P_n = [x_n, y_n];$$

3. kończy obchód w punkcie początkowym, tzn. przyjmuje $P_{n+1} = P_1$.

Przykład

Poniższy obrazek przedstawia działkę ($n = 5$) oraz kolejne punkty pomiaru zanotowane przez geodetę:

$$P_1 = [2,2], \quad P_2 = [1,7], \quad P_3 = [9,8], \quad P_4 = [10,4], \quad P_5 = [8,2], \quad P_6 = P_1.$$



Następnie pan G wyznacza pole działki za pomocą wzoru

$$pole = \frac{1}{2} \sum_{i=1}^n (x_{i+1} - x_{i-1}) \cdot y_i,$$

gdzie x_0 przyjmuje jako x_n .

Obliczanie pola z podanego wzoru realizowane jest według następującego algorytmu:

Dane:

n — liczba całkowita, $n \geq 0$,

$x_1, y_1; x_2, y_2; \dots; x_n, y_n$ — współrzędne punktów pomiaru.

Wynik:

pole działki

Algorytm:

$$x_0 \leftarrow x_n$$

$$x_{n+1} \leftarrow x_1$$

$$y_{n+1} \leftarrow y_1$$

$$pole = 0$$

dla $i = 1, 2, \dots, n$ wykonuj

$$pole \leftarrow pole + (x_{i+1} - x_{i-1}) \cdot y_i$$

zwróć $\frac{pole}{2}$ i zakończ

14.1.

Przeanalizuj działanie powyższego algorytmu dla przykładu z rysunku i uzupełnij poniższą tabelkę.

i	x_i	y_i	$x_{i+1} - x_{i-1}$	$(x_{i+1} - x_{i-1}) \cdot y_i$
1	2	2		
2	1	7		
3	9	8		
4	10	4		
5	8	2		

Podaj obliczone pole działki:

14.2.

Niech punkty $A = [x_1, y_1]$, $B = [x_2, y_2]$, $C = [x_3, y_3]$ będą wierzchołkami pewnego trójkąta, podanymi w kolejności zgodnej z ruchem wskazówek zegara. Bazując na metodzie geodety, podaj wzór na pole trójkąta ABC .

14.3.

Wyraź wzorem liczbę operacji mnożenia oraz łączną liczbę operacji dodawania i odejmowania współrzędnych punktów, jaka zostanie wykonana przez podany algorytm podczas obliczania pola powierzchni działki o kształcie wielokąta n -wierzchołkowego.

Uzupełnij poniższą tabelkę:

Działanie	Liczba operacji
dodawanie i odejmowanie	
mnożenie	

Zadanie 15.

Wiązka zadań Zbiór Cantora

Zbiór (fraktal) Cantora rzędu 0 jest równy odcinkowi jednostkowemu $[0; 1]$. Zbiór Cantora rzędu 1 uzyskujemy, dzieląc odcinek jednostkowy $[0; 1]$ na trzy równe części i usuwając odcinek środkowy (pozostawiając zaś jego końce). Składa się on zatem z dwóch fragmentów odcinka jednostkowego: $[0; 1/3]$, $[2/3; 1]$.

Ogólnie zbiór Cantora rzędu $n+1$ tworzymy ze zbioru Cantora rzędu n w następujący sposób: każdy odcinek zbioru Cantora rzędu n dzielimy na trzy równe części i usuwamy odcinek środkowy, pozostawiając jego końce. Zgodnie z tą regułą zbiór Cantora rzędu 2 składa się z odcinków: $[0; 1/9]$, $[2/9; 3/9]$, $[6/9; 7/9]$ i $[8/9; 1]$.



Rysunek. Geometryczna ilustracja zbiorów Cantora rzędu 0, 1, 2, 3 i 4

15.1.

Uzupełnij poniższą tabelę, podając liczbę odcinków w zbiorach Cantora podanych rzędów.

n	Liczba odcinków w zbiorze Cantora rzędu n
0	1
1	2
2	4
3	8
5	
6	
9	
10	

Podaj ogólny wzór określający $C(n)$, liczbę odcinków w zbiorze Cantora rzędu n :

$$C(n) = \dots$$

15.2.

Zauważ, że każdy odcinek w zbiorze Cantora ustalonego rzędu ma tę samą długość. Uzupełnij poniższą tabelę, podając długość jednego odcinka w zbiorach Cantora podanych rzędów.

n	Długość jednego odcinka w zbiorze Cantora rzędu n
0	1
1	1/3
2	1/9
3	1/27
4	
5	
6	
7	

Podaj ogólny wzór określający długość $D(n)$ jednego odcinka w zbiorze Cantora rzędu n :

$$D(n) = \dots$$

15.3.

Uzupełnij poniższą listę odcinków zbioru Cantora rzędu 3, których końce zapisane są jako ułamki zwykłe **nieskracalne**:

[0; 1/27], [2/27; 1/9],

Końce odcinków zbiorów Cantora można opisać w zwarty i regularny sposób w systemie trójkowym. W szczególności odcinki zbioru Cantora rzędu 1 zapisane w systemie trójkowym to

[0; 0,1], [0,2; 1],

natomiast odcinki zbioru Cantora rzędu 2 zapisane w systemie trójkowym to

[0; 0,01], [0,02; 0,1], [0,2; 0,21] i [0,22; 1].

Podaj poniżej odcinki zbiorów Cantora rzędu 3 zapisane w systemie o podstawie 3 (trójkowym):

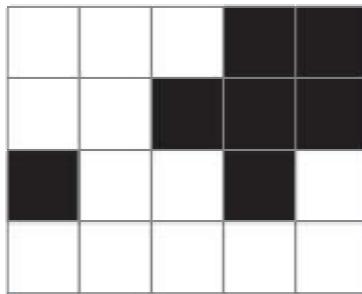
[0; 0,001],
.....

Zadanie 16.

Wiązka zadań *Odległość na planszy*

Dana jest prostokątna plansza o rozmiarach m na n (m wierszy, n kolumn), podzielona na $m \cdot n$ pól — jednostkowych kwadratów. Przez *pole* (i, j) rozumiemy pole, które jest w i -tym wierszu oraz j -tej kolumnie. Wiersze numerujemy kolejno od 1 do m , z góry do dołu, a kolumny kolejno od 1 do n , od lewej do prawej. **Dwa pola uważaemy za sąsiednie, jeśli mają wspólny bok.**

Niektóre pola tej planszy zostały wycięte — na rysunku oznaczone są na czarno.



Poniżej podany jest algorytm, który tworzy tablicę $D[1..m, 1..n]$ (o m wierszach oraz n kolumnach), odpowiadającą planszy, a następnie wpisuje liczby całkowite do komórek tablicy odpowiadających niewyciętym (białym) polom na planszy. Każda wpisana liczba mierzy w pewien sposób odległość odpowiedniego pola od lewego górnego rogu planszy (pola (1,1) — załóż, że to pole nigdy nie jest wycięte). Przeanalizuj algorytm i rozwiąż podanej niżej zadania.

dla każdego niewyciętego pola (i, j)

$D[i, j] \leftarrow -1$

$D[1, 1] \leftarrow 0$

$k \leftarrow 0$

powtarzaj:

jeśli nie ma takich pól (x, y) , że $D[x, y] = k$

zakończ algorytmu

dla każdego pola (x, y) takiego, że $D[x, y] = k$ **wykonaj**

dla każdego pola (a, b) sąsiadującego z (x, y) **wykonaj**

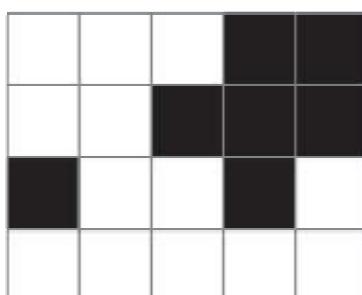
jeśli (a, b) nie jest wycięte **oraz** $D[a, b] = -1$

$D[a, b] \leftarrow k+1$

$k \leftarrow k+1$

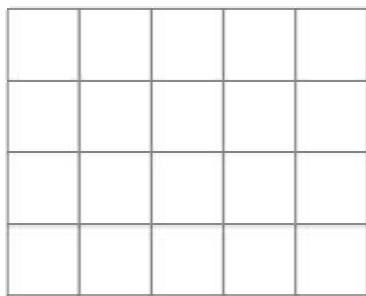
16.1.

Dla podanej poniżej planszy wyznacz wartości, które znajdą się w tablicy D po zakończeniu wykonywania algorytmu. Wpisz te wartości w odpowiednie pola planszy:



16.2.

Zaprojektuj taką planszę o rozmiarach 4·5, dla której w wynikowej tablicy D pojawi się liczba 10. Odpowiedź podaj, kolorując na poniższym rysunku pewne pola na czarno oraz wpisując liczbę „10” w odpowiednim polu:



16.3.

Zaprojektuj taką planszę (wybranych przez siebie rozmiarów), dla której w wynikowej tablicy D po zakończeniu wykonywania algorytmu w pewnej komórce pozostanie liczba -1. Narysuj taką planszę i wpisz -1 na wszystkich białych polach, na których ta wartość pozostanie do końca działania algorytmu.

Zadanie 17.

Wiązka zadań System hamowania

Nowoczesne samochody mają wbudowany komputer z systemem wspomagającym prowadzenie pojazdu. Jedną z jego funkcji jest aktywacja systemu hamowania w momencie, gdy samochód znajdzie się zbyt blisko pojazdu znajdującego się przed nim.

Automatyczne hamowanie wykorzystuje czujniki, które mierzą prędkość samochodu oraz jego odległość od pojazdu znajdującego się przed nim. Reakcja systemu zależy od prędkości samochodu, w którym jest on uruchomiony. W zadaniach ograniczamy się do analizy działania systemu przy małych prędkościach samochodu.

Przyjmujemy, że automatyczne uruchomienie hamowania następuje, jeśli odległość od najbliższego pojazdu jest **mniejsza niż** 15 metrów oraz prędkość samochodu, w którym działa system, wynosi **mniej niż** 30 km/h (zobacz poniższy rysunek). Po automatycznym uruchomieniu hamowania prędkość samochodu będzie spadać, a hamulce będą włączone do momentu osiągnięcia przez samochód prędkości 0 km/h, niezależnie od tego, jak będzie się zmieniać odległość od najbliższego pojazdu.

W analizie działania systemu hamowania przyjmuje się, że samochód nie przekracza prędkości 350 km/h.



Działanie systemu automatycznego hamowania zostało poniżej błędnie opisane w pseudokodzie.

odczytaj prędkość samochodu;
odczytaj odległość od najbliższego pojazdu;
jeżeli prędkość samochodu < 30 lub odległość od najbliższego pojazdu < 15:
dopóki prędkość samochodu = 100 wykonuj
wyślij polecenie hamowania do układu hamulcowego;
odczytaj prędkość samochodu

17.1.

W powyższym pseudokodzie występują dwa błędy powodujące niezgodność z podanym wcześniej opisem. Znajdź i popraw oba błędy.

Błędny zapis w pseudokodzie	Poprawny zapis w pseudokodzie

17.2.

Przyjmujemy, że system hamowania odczytuje prędkość samochodu i odległość od najbliższego pojazdu na tyle często, że sąsiednie pomiary mogą się różnić co najwyżej o odpowiednio 1 km/h i 1 m (pomiary prędkości podawane są w km/h, w zaokrągleniu do wartości całkowitych; pomiary odległości podawane są w m, również w zaokrągleniu do wartości całkowitych). Oceń, czy podane w poszczególnych wierszach poniższej tabeli dane testowe są danymi standardowym (niezmieniające stanu systemu hamowania), brzegowymi (wartości, które mogą powodować włączenie lub wyłączenie systemu hamowania) czy też danymi niezgodnymi ze specyfikacją.

Dane testowe	Typ danych testowych
prędkość samochodu — 29 km/h, odległość między samochodami — 1 m	brzegowe
prędkość samochodu — 400 km/h, odległość od najbliższego pojazdu — 0 m	
prędkość samochodu — 13 km/h, odległość od najbliższego pojazdu — 8 m	
prędkość samochodu — 45 km/h, odległość od najbliższego pojazdu — 17 m	
prędkość samochodu — 0 km/h, odległość od najbliższego pojazdu — 17 m	

17.3.

Wskaż, które z poniższych dokończeń zdań są prawdziwe (P), a które fałszywe (F), wstawiając znak X w odpowiedniej kolumnie.

Celem testowania systemu dla różnych typów danych jest

	P	F
upewnienie się, że system poradzi sobie z odczytem parametrów z czujników i tempo jego reakcji pozwoli uniknąć zderzenia.		
zwiększenie precyzji pomiarów prędkości i odległości.		
zmniejszenie długości kodu programu.		
sprawdzenie, czy system działa zgodnie z podaną specyfikacją.		

17.4.

Uzupełnij poniższą tabelę, wpisując w ostatniej kolumnie słowo *włączony*, *wyłączony* (gdy automatyczne hamowanie będzie przy danych odczytach włączone/wyłączone niezależnie od wartości poprzedniego pomiaru) lub *nieustalony* (gdy aktywność automatycznego hamowania zależy zarówno od bieżącego, jak i od poprzedniego pomiaru).

Prędkość samochodu	Odległość między samochodami	Stan automatycznego hamowania
poniżej 30 km/h	poniżej 15m	włączony
poniżej 30 km/h	równa 15m	
równa 30 km/h	poniżej 15m	
równa 30 km/h	równa 15m	
powyżej 30 km/h	dowolna	

1.2. Tworzenie algorytmów

Zadanie 18.

Wiązka zadań Największy wspólny dzielnik

Opisana poniżej funkcja wyznacza największy wspólny dzielnik (NWD) dwóch liczb całkowitych dodatnich.

Specyfikacja

Dane:

liczby całkowite dodatnie a i b .

Wynik:

Największy wspólny dzielnik liczb a i b .

funkcja $NWD(a, b)$

dopóki $b \neq 0$ wykonuj

$$r \leftarrow a \bmod b \quad (*)$$

$$a \leftarrow b$$

$$b \leftarrow r$$

zwróć a i zakończ

18.1.

Przeanalizuj działanie funkcji NWD i dla wskazanych argumentów podaj liczbę wykonan operacji modulo:

a	b	liczba operacji mod (*)
25	15	
116	324	
762	282	

18.2.

Wykorzystując funkcję NWD oraz następującą zależność:

$$NWD(a_1, a_2, \dots, a_n) = NWD(NWD(a_1, a_2, \dots, a_{n-1}), a_n),$$

możemy wyznaczyć największy wspólny dzielnik n liczb całkowitych dodatnich a_1, a_2, \dots, a_n .

Przykład

$$NWD(15, 24, 60) = NWD(NWD(15, 24), 60) = NWD(3, 60) = 3.$$

Uzupełnij poniższą tabelkę i dla wskazanych n liczb całkowitych dodatnich a_1, a_2, \dots, a_n oblicz ich największy wspólny dzielnik.

a_1, a_2, \dots, a_n	$NWD(a_1, a_2, \dots, a_n)$
36, 24, 72, 150	
119, 187, 323, 527, 731	
121, 330, 990, 1331, 110, 225	

18.3.

Napisz algorytm, który korzystając z funkcji $NWD(a, b)$, wyznaczy największy wspólny dzielnik ciągu liczb a_1, a_2, \dots, a_n .

Dane:

a_1, a_2, \dots, a_n — liczby całkowite dodatnie.

Wynik:

Największy wspólny dzielnik liczb a_1, a_2, \dots, a_n .

Komentarz do zadania

18.1.

Dla pierwszego przykładu: $a = 25$, $b = 15$, kolejne reszty r są równe: 10, 5, 0, zatem wykonywane są 3 operacje $a \bmod b$. Dla $a = 116$, $b = 324$ kolejne reszty r są równe: 116, 92, 24, 20, 4, 0, stąd w pętli wykonywanych jest 6 operacji $a \bmod b$. Podobnie jest dla $a = 762$, $b = 282$: zmienna r przyjmuje kolejno następujące wartości: 198, 84, 30, 24, 6, 0, stąd wykonywanych jest 6 operacji $a \bmod b$.

18.2.

W tym zadaniu należy przeanalizować obliczanie NWD dla n liczb z wykorzystaniem funkcji napisanej w treści zadania:

$$\text{NWD}(36, 24, 72, 150) =$$

$$\text{NWD}(\text{NWD}(36, 24, 72), 150) =$$

$$\text{NWD}(\text{NWD}(\text{NWD}(36, 24), 72), 150) =$$

$$\text{NWD}(\text{NWD}(12, 72), 150) =$$

$$\text{NWD}(12, 150) = 6$$

$$\text{NWD}(119, 187, 323, 527, 731) =$$

$$\text{NWD}(\text{NWD}(119, 187, 323, 527), 731) =$$

$$\text{NWD}(\text{NWD}(\text{NWD}(17, 323), 527), 731) =$$

$$\text{NWD}(\text{NWD}(17, 527), 731) =$$

$$\text{NWD}(17, 731) = 17$$

$$\text{NWD}(121, 330, 990, 1331, 110, 225) =$$

$$\text{NWD}(\text{NWD}(121, 330, 990, 1331, 110), 225) =$$

$$\text{NWD}(\text{NWD}(\text{NWD}(121, 330, 990, 1331), 110), 225) =$$

$$\text{NWD}(\text{NWD}(\text{NWD}(\text{NWD}(121, 330), 990), 1331), 110), 225) =$$

$$\text{NWD}(\text{NWD}(\text{NWD}(\text{NWD}(121, 330), 990), 1331), 110), 225) =$$

$$\text{NWD}(\text{NWD}(\text{NWD}(\text{NWD}(11, 990), 1331), 110), 225) =$$

$$\text{NWD}(\text{NWD}(\text{NWD}((11, 1331), 110), 225) =$$

$$\text{NWD}(\text{NWD}(11, 110), 225) =$$

$$\text{NWD}(11, 225) = 1$$

18.3.

Algorytm utworzymy w oparciu o tożsamość:

$$\text{NWD}(a_1, a_2, \dots, a_n) = \text{NWD}(\text{NWD}(a_1, a_2, \dots, a_{n-1}), a_n),$$

podaną w zadaniu drugim.

Za największy wspólny dzielnik obieramy liczbę a_1 , a następnie iteracyjnie obliczamy największy wspólny dzielnik zapamiętanego wyniku oraz liczby a_i dla kolejnych $i = 2, 3, 4, \dots, n$. Prowadzi to do następującego rozwiązania zadania:

$$w \leftarrow a_1$$

dla $i = 2, 3, 4, \dots, n$ **wykonuj**

$$w \leftarrow \text{NWD}(w, a_i)$$

zwróć w **i zakończ**

Podane rozwiązanie jest poprawne także wtedy, gdy n jest równe 1.

Rozwiążanie

18.1.

a	b	liczba operacji mod
25	15	3
116	324	6
762	282	6

18.2.

a_1, a_2, \dots, a_n	$\text{NWD}(a_1, a_2, \dots, a_n)$
36, 24, 72, 150, 114	6
119, 187, 323, 527, 731	17
121, 330, 990, 1331, 110, 225	1

18.3.

Przykładowa poprawna odpowiedź:

```
w ← a1
dla i = 2, 3, 4, ..., n wykonuj
    w ← NWD (w, ai)
zwróć w i zakończ
```

Zadanie 19.

Wiązka zadań *Zakupy międzyplanetarne*

Mieszkańcy galaktyki Różnoliczbowo zamieszkują 9 planet: Liczbowo₂, Liczbowo₃, ..., Liczbowo₁₀. Na każdej planecie Liczbowo_i jej mieszkańcy posługują się systemem liczbowym o podstawie i . Na każdej planecie wszystkie ceny są liczbami naturalnymi.

19.1.

Mieszkańcy czterech sąsiadujących planet: Liczbowo₂, Liczbowo₄, Liczbowo₈ oraz Liczbowo₁₀ często podróżują pomiędzy tymi planetami i kupują różne towary. W poniższej tabeli znajdują się ceny wybranych towarów zakupionych przez jedną osobę na różnych planetach. Uzupełnij tabelę, przeliczając **podane** ceny na systemy liczbowe wszystkich czterech planet.

Towar	Cena towaru zapisana w systemie liczbowym planety			
	Liczbowo ₂	Liczbowo ₄	Liczbowo ₈	Liczbowo ₁₀
Kozaki	10111011			
Płaszcz			724	
Skuter				1458

19.2.

Na różnych planetach ten sam towar może mieć różną cenę, na przykład cena ciasta kokosowego na planecie Liczbowo₁₀ wynosi 38₁₀, zaś na planecie Liczbowo₈ jego cena wynosi 55₈ (równą 45₁₀).

Mieszkańcy planety Liczbowo₁₀ są bardzo oszczędni i przed zakupami porównują ceny towarów na wybranych planetach. Uzupełnij w poniższej tabeli relacje (>, <, =) pomiędzy poszczególnymi cenami.

Liczbowo _x	Relacja	Liczbowo _y
110000100 ₂	>	556 ₈
3123 ₄		1747 ₈
110 ₁₀		11010 ₃
266 ₉		110100 ₃
110111101 ₂		674 ₈

19.3.

Sprzedawcy sklepów planety Liczbowo_i wyliczają wartość zakupów klientów, sumując ceny zakupionych towarów w systemie obowiązującym na ich planecie; stosują przy tym metodę dodawania pisemnego.

Przykłady

Dodawanie w systemie o podstawie 2:

$$\begin{array}{r}
 & 1 & 0 & 1 & 0 \\
 1 & 0 & 0 & 1 & 1 \\
 & 1 & 1 & 0 & 1 \\
 +_2 & 1 & 1 & 0 & 0 \\
 \hline
 1 & 0 & 0 & 1 & 1
 \end{array}$$

Dodawanie w systemie o podstawie 4:

$$\begin{array}{r}
 & 2 & 2 \\
 1 & 0 & 3 \\
 & 3 & 1 \\
 +_4 & 1 & 2 \\
 \hline
 1 & 0 & 0 & 3
 \end{array}$$

Podsumuj rachunki pana Dwójkowskiego (z planety Liczbowo₂) oraz pana Czwórkowskiego (z planety Liczbowo₄).

Rachunek pana Dwójkowskiego:

$$\begin{array}{r}
 & 1 & 0 & 1 & 1 \\
 1 & 0 & 0 & 1 & 0 \\
 1 & 1 & 0 & 1 & 1 \\
 & 1 & 1 & 0 & 1 & 0 \\
 & 1 & 1 & 0 & 0 & 1 \\
 +_2 & 1 & 0 & 1 & 0 & 1 \\
 \hline
 \text{SUMA} & & & & &
 \end{array}$$

Rachunek pana Czwórkowskiego:

$$\begin{array}{r} & & 3 & 3 \\ & 1 & 0 & 2 \\ & 3 & 1 & 1 \\ & 1 & 3 & 1 \\ & 1 & 2 & 3 \\ +_4 & & 3 & 0 & 1 \\ \hline \text{SUMA} & & \dots & \dots & \dots \end{array}$$

Podaj różnicę wartości obu rachunków w systemie obowiązującym na planecie Liczbowo₁₀.

Różnica rachunków w systemie obowiązującym na planecie Liczbowo₁₀ wynosi:

19.4.

Właściciele sieci sklepów ulokowanych na wszystkich planetach galaktyki postanowili dostarczyć do sklepów kalkulatory, które będą dodawały ceny w opisany powyżej sposób. Aby pomóc właścicielom, podaj algorytm dodawania (w postaci pseudokodu lub w języku programowania), który dla dwóch liczb a i b zapisanych w systemie o podstawie $p \in [2,9]$ wyznacza i wypisuje wartość sumy $a +_p b$ zapisaną w systemie o podstawie p zgodnie z poniższą specyfikacją. Twój algorytm **nie może** dokonywać zamiany liczb a i b na inny system pozycyjny.

Specyfikacja

Dane:

p — podstawa systemu, liczba naturalna z przedziału $[2,9]$,

n — liczba cyfr liczb naturalnych a , $b \leq 255$ (przyjmujemy, że krótsza liczba jest uzupełniona z lewej strony zerami, tak aby obie liczby miały taką samą długość),

$A[n], A[n-1], \dots, A[1]$ — kolejne cyfry liczby a zapisanej w systemie o podstawie p ,

$B[n], B[n-1], \dots, B[1]$ — kolejne cyfry liczby b zapisanej w systemie o podstawie p ,

Wyniki:

wartość liczby $c = a +_p b$ zapisana w systemie o podstawie p w postaci ciągu cyfr $C[n+1], C[n], \dots, C[1]$.

Przykład

Dla następujących danych:

$$p = 4$$

$$n = 4,$$

$$\text{Liczba } a = 3122_4$$

$$\text{Liczba } b = 21_4$$

$$\text{Wynikiem jest liczba } c = 3203_4$$

Zawartość tablic A, B, C:

i	5	4	3	2	1
A[i]		3	1	2	2
B[i]		0	0	2	1
C[i]	0	3	2	0	3

Komentarz do zadania

19.1.

Powszechnie znane są algorytmy konwersji liczby z zapisu dziesiętnego na zapis w systemie pozycyjnym o podstawie $s < 10$ i odwrotnie: z zapisu w systemie o podstawie s na system dziesiętny. Aby uniknąć wielokrotnego wykonywania takich konwersji, skorzystamy z zależności między reprezentacjami liczb w systemie o podstawie 2, podstawi 4 = 2·2 oraz podstawi 8=2·2·2. Skoncentrujmy się najpierw na systemach o podstawach 2 i 4:

- reprezentację liczby w systemie czwórkowym można uzyskać z jej reprezentacji w systemie binarnym (czyli o podstawie 2), wybierając od końca pary cyfr i zamieniając je na ich czwórkowe reprezentacje;
- reprezentację liczby w systemie binarnym można uzyskać z jej reprezentacji w systemie czwórkowym, zamieniając każdą cyfrę czwórkową na jej dwucyfrową reprezentację binarną.

Ponieważ $8=2^3$, analogiczna własność zachodzi dla konwersji między systemem binarnym a systemem ósemkowym, z tą różnicą, że zamiast bloków 2 cyfr rozważamy bloki o długości 3. Korzystając z powyższych obserwacji, uzyskujemy rozwiązanie:

Towar	Cena towaru zapisana w systemie liczbowym planety			
	Liczbowo ₂	Liczbowo ₄	Liczbowo ₈	Liczbowo ₁₀
Kozaki	10111011	2323	273	187
Płaszcz	111010100	13110	724	468
Skuter	10110110010	112302	2662	1458

Dokładniej, dla $10111011_{(2)}$ uzyskujemy:

- reprezentację czwórkową: dzieląc 10111011 na bloki 10, 11, 10, 11 i zapisując w systemie o podstawie 4 wartość każdego bloku: 2, 3, 2 i 3;

1	0	1	1	1	0	1	1
10	11	10	11				
2	3	2	3				

- reprezentację ósemkową: dzieląc 10111011 na bloki 10, 111, 011 i zapisując w systemie o podstawie 4 wartość każdego bloku: 2, 7 i 3.

1	0	1	1	1	0	1	1
010	111	011					
2	7	3					

Z kolei z $724_{(8)}$ uzyskujemy:

- reprezentację binarną: zamieniając każdą cyfrę ósemkową na jej 3-cyfrową reprezentację binarną, czyli 111, 010, 100; daje to reprezentację 111010100;
- reprezentację czwórkową: dzieląc binarną reprezentację 111010100 na bloki 01, 11, 01, 01 i 00, zapisując w systemie o podstawie 4 wartość każdego bloku: 1, 3, 1, 1 i 0.

Znając zasady konwersji między systemami o podstawach 2, 4 i 8, reprezentację dziesiętną możemy uzyskać za pomocą standardowego algorytmu zamiany liczby z systemu o podstawie p różnej od 10 (np. $p = 2$) na system dziesiętny. Analogicznie, znając reprezentację dziesiętną,

wystarczy znaleźć jej reprezentację w jednym z pozostałych systemów, a potem zastosować omówione powyżej reguły konwersji między systemami o podstawach 2, 4 i 8.

19.2.

Używając standardowych algorytmów zamiany z systemu niedziesiętnego na dziesiętny, możemy wszystkie liczby przekształcić na postać dziesiętną i wówczas porównać. Metoda ta wymaga dość dużo obliczeń, dlatego pokażemy sposób wymagający mniej pracy. Podobnie jak w rozwiązaniu zadania 1 polega on będzie na wykorzystaniu faktu, że dwie cyfry w systemie o podstawie s odpowiadają jednej cyfrze w systemie o podstawie s^2 , podobnie trzy cyfry w systemie o podstawie s odpowiadają jednej cyfrze w systemie o podstawie s^3):

$$556_8 = 101101110_2 < 110000100_2$$

$$3123_4 = 11011011_2, \quad 1747_8 = 00111100111_2, \text{ czyli } 3123_4 < 1747_8$$

$$266_9 = 022020_3, \text{ czyli } 266_9 < 110100_3$$

$$674_8 = 110111100_2, \text{ czyli } 674_8 < 1101111012$$

Jedyny przykład wymagający konwersji na system dziesiętny to porównanie 110_{10} i $11010_3 = 81_{10} + 27_{10} + 3_{10} = 111_{10}$.

19.3.

Możemy wszystkie liczby w rachunku zamienić na system dziesiętny i je dodawać. Inne rozwiązanie polega na zastosowaniu metody dodawania pisemnego przeprowadzonego na reprezentacji binarnej i czwórkowej. Pokażemy je dla par liczb z rachunku pana Dwójkowskiego. Najpierw dodamy dwie pierwsze liczby.

przeniesienie:		0	0	1	0	
			1	0	1	1
+ ₂	1	0	0	1	0	0
SUMA:		1	1	1	0	1

Następnie do wyniku dodamy trzecią liczbę

przeniesienie:		1	1	1	1	1
			1	1	1	0
+ ₂	1	1	0	1	1	1
SUMA:		1	0	1	0	0

Dodając następnie liczby: 11010_2 , 11001_2 i 101011_2 , uzyskamy wynik 10110010_2 .

Analogicznie postępujemy dla reprezentacji czwórkowych. Suma dwóch pierwszych liczb to:

przeniesienie:			1	1		
				3	3	
+ ₄	1	0	2			
SUMA:		2	0	1		

Dodając następnie liczby: 311_4 , 131_4 , 123_4 i 301_4 , uzyskamy wynik 2333_4 .

Następnie zamieniamy obie obliczone sumy na system dziesiętny:

$$10110010_2 = 2_{10} + 16_{10} + 32_{10} + 128_{10} = 178_{10} \text{ oraz } 2333_4 = 3_{10} + 3_{10} \cdot 4_{10} + 3 \cdot 16_{10} + 2 \cdot 64_{10} = 191_{10}.$$

A zatem różnica wartości rachunków wynosi 13_{10} .

19.4.

Zadanie można rozwiązać, implementując zasadę dodawania pisemnego, której przykłady podaliśmy w rozwiązaniu zadania 3. Zgodnie z tą zasadą dodajemy odpowiadające sobie cyfry i przeniesienia, zaczynając od prawej strony. Przyjmujemy przy tym, że ostatnie przeniesienie $R[1]$ (na skrajnie prawej pozycji 1) jest równe zero. Ponadto wiemy, że:

- i -ta cyfra wyniku jest równa $(A[i] + B[i] + R[i]) \bmod p$, gdzie $R[i]$ to i -te przeniesienie,
- $(i+1)$ -sze przeniesienie $R[i+1]$ jest równe $(A[i] + B[i] + R[i]) \div p$,

gdzie mod i div oznaczają odpowiednio operacje reszty z dzielenia i wyniku dzielenia całkowitego (tzn. zaokrąglenia dokładnego wyniku dzielenia do liczby całkowitej w dół). Musimy jednak uwzględnić, że wynik może być o jedną cyfrę dłuższy od dodawanych liczb. Dlatego przyjmiemy, że wynik ma $n+1$ cyfr i najbardziej znaczącą cyfrę wyniku zapiszemy na pozycji $n+1$.

```
i ← 1  
R[1] ← 0  
dopóki  $i < n+1$  wykonuj  
    c ←  $A[i] + B[i] + R[i]$   
    C[i] ← c mod p  
    R[i+1] ← c div p  
    i ← i + 1  
C[n+1] ← R[n+1]
```

W modelu odpowiedzi zamieściliśmy trochę inne rozwiązanie:

- zamiast tablicy przeniesień R przechowujemy tylko aktualne przeniesienie, w zmiennej r ;
- pokazujemy tam, jak można uniknąć stosowania operatorów mod i div, korzystamy przy tym z tego, że $A[i]+B[i]$ jest zawsze nie większe niż $2p - 2$, a reszta $R[i]$ jest równa 0 lub 1.

Zadanie 20.

Wiązka zadań *Liczba narcystyczna*

Niech dana będzie liczba naturalna x , której zapis dziesiętny ma n cyfr:

$$x = a_{n-1}10^{n-1} + a_{(n-2)}10^{n-2} + \cdots + a_110 + a_0 \quad (a_{n-1} \neq 0).$$

Powiemy, że liczba x jest *narcystyczna*, jeśli suma jej cyfr podniesionych do potęgi n -tej jest równa x , tzn.

$$a_{n-1}^n + a_{n-2}^n + \cdots + a_1^n + a_0^n = x.$$

Na przykład liczba 1634 jest narcystyczna, ponieważ

$$1^4 + 6^4 + 3^4 + 4^4 = 1634.$$

Powiemy, że liczba x jest *B-narcystyczna*, jeśli jej zapis w systemie o podstawie B ma n cyfr, których suma n -tych potęg jest równa x , tzn.

$$x = a_{n-1}B^{n-1} + a_{(n-2)}B^{n-2} + \cdots + a_1B + a_0 \quad \text{oraz} \quad a_{n-1}^n + a_{n-2}^n + \cdots + a_1^n + a_0^n = x.$$

Na przykład liczba 289 jest 5-narcystyczna, ponieważ

$$289 = (2124)_5 = 2 \cdot 5^3 + 1 \cdot 5^2 + 2 \cdot 5 + 4 \text{ oraz } 289 = 2^4 + 1^4 + 2^4 + 4^4.$$

20.1.

Uzupełnij brakujące cyfry tak, aby powstałe liczby były liczbami narcystycznymi.

31

40

5448

20.2.

Sprawdź, które z poniższych liczb (podanych w systemie dziesiętnym) są B -narcystyczne dla podanych wartości B . Uzupełnij poniższą tabelkę, wpisując *prawda* lub *fałsz* w zależności od tego, czy liczba jest B -narcystyczna, czy też nie.

x	B	<i>prawda/fałsz</i>
3433	6	
4890	5	
8956	3	
15345	2	

20.3.

Napisz algorytm (w pseudokodzie lub wybranym języku programowania), który sprawdza, czy dana liczba x jest B -narcystyczna.

Dane:

x — liczba całkowita, $x \geq 0$,

B — liczba całkowita, $B \geq 2$.

Wynik:

TAK, jeśli liczba x jest B -narcystyczna, NIE — w przeciwnym przypadku.

Zadanie 21.

Wiązka zadań *Szybkie podnoszenie do potęgi*

W algorytmach szybkiego potęgowania można wykorzystać binarną reprezentację wykładnika dla obliczenia wartości x^k , gdzie k jest liczbą naturalną, $k \neq 0$, zaś x jest liczbą rzeczywistą. Przyjmijmy, że binarnym rozwinięciem wykładnika k jest ciąg $(k_n k_{n-1} k_{n-2} \dots k_2 k_1 k_0)_2$.

Jedna z metod wyznaczania x^k polega na obliczaniu potęg liczby x dla wykładników o binarnych reprezentacjach:

$k_n,$

$k_n k_{n-1},$

$k_n k_{n-1} k_{n-2},$

$\dots,$

$k_n k_{n-1} k_{n-2} \dots k_1,$

$k_n k_{n-1} k_{n-2} \dots k_1 k_0,$

Inaczej mówiąc, uwzględniamy coraz dłuższe fragmenty ciągu $(k_n k_{n-1} k_{n-2} \dots k_2 k_1 k_0)_2$. W pierwszym kroku przyjmujemy, że wynik jest równy x , gdyż $k_n = 1$. Znając wartość x do potęgi o binarnym zapisie $(k_n k_{n-1} k_{n-2} \dots k_i)_2$, możemy łatwo wyliczyć x do potęgi o binarnym zapisie $(k_n k_{n-1} k_{n-2} \dots k_i k_{i-1})_2$: podnosimy dotychczasowy wynik do kwadratu (do czego wystarczy jedno mnożenie). Jeśli $k_{i-1} = 1$, dodatkowo mnożymy uzyskany wynik przez x .

Przykład

Niech $k = 13 = (1101)_2$. Kolejne wyznaczane w naszym algorytmie potęgi to:

$$x^1, x^3 = x^2 \cdot x, x^6 = (x^3)^2, x^{13} = (x^6)^2 \cdot x,$$

zaś liczba wykonanych mnożeń jest równa 5 (zauważ, że aby obliczyć x^3 , musisz najpierw obliczyć x^2 , a aby obliczyć x^2 , musisz wykonać jedno mnożenie: $x^2 = x \cdot x$).

21.1.

Korzystając z przedstawienia wykładnika w postaci binarnej, podaj kolejne potęgi liczby x wyznaczane powyższą metodą przy obliczaniu x^{38} .

21.2.

Uzupełnij tabelkę. Oblicz, ile mnożeń wykonywanych jest dla kolejnych wykładników.

k	reprezentacja binarna k	liczba mnożeń
4	100	2
5	101	3
6		
7		
8		
15		
16		
22		
32		

21.3.

W wybranej przez siebie notacji (lista kroków, pseudokod, język programowania) napisz algorytm, który dla zadanej binarnej reprezentacji liczby naturalnej k , $k \neq 0$, oraz rzeczywistej liczby x oblicza wartość x^k zgodnie z metodą opisaną na początku zadania.

Specyfikacja

Dane:

x — liczba rzeczywista,

n — liczba całkowita nieujemna,

$k_n k_{n-1} k_{n-2} \dots k_1 k_0$ — ciąg tworzący binarną reprezentację wykładnika k ,

Wynik:

liczba rzeczywista $p = x^k = \underbrace{x \cdot x \cdot \dots \cdot x}_{k \text{ razy}}$

Zadanie 22.

Wiązka zadań Schemat Hornera

Schemat Hornera jest bardzo efektywną metodą obliczania wartości wielomianu

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0,$$

gdzie dane liczby rzeczywiste a_0, a_1, \dots, a_n nazywamy *współczynnikami*, a liczba całkowita $n \geq 0$ oznacza *stopień* wielomianu.

Schemat bazuje na zależności

$$P(x) = x(a_n x^{n-1} + a_{n-1} x^{n-2} + \cdots + a_2 x^1 + a_1) + a_0 = x \cdot Q(x) + a_0,$$

gdzie

$$Q(x) = a_n x^{n-1} + a_{n-1} x^{n-2} + \cdots + a_2 x^1 + a_1.$$

Stąd otrzymujemy następujący schemat obliczania wartości $P(x)$:

Dane:

- n — liczba całkowita, $n \geq 0$,
- x — liczba rzeczywista,
- a_0, a_1, \dots, a_n — liczby rzeczywiste.

Wynik:

wartość $P(x)$

Algorytm (schemat Hornera):

```

 $w \leftarrow a_n$ 
dla  $k = n - 1, n - 2, \dots, 0$  wykonuj
(*)       $w \leftarrow x \cdot w + a_k$ 
zwróć  $w$  i zakończ

```

22.1.

Uzupełnij poniższą tabelkę, podając wartości danych, jakie należy przyjąć w powyższym schemacie, aby wyznaczyć wartość $P(6)$ dla wielomianu

$$P(x) = 10x^5 - 13x^4 + x^3 + 2x^2 - 8x + 7.$$

Dane	Wartości
liczba naturalna n	
liczba rzeczywista x	
liczby rzeczywiste a_0, a_1, \dots, a_n	

22.2.

Uzupełnij poniższą tabelkę, wyrażając wzorem liczbę operacji mnożenia i dodawania, jaka zostanie wykonana przez schemat Hornera (w wierszu oznaczonej przez (*)) dla danego wielomianu

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0.$$

Działanie	Liczba operacji
Dodawanie	
Mnożenie	

22.3.

Przeanalizuj działanie schematu Hornera podczas obliczania wartości $P(2)$ dla wielomianu

$$P(x) = 4x^6 - 3x^5 + 2x^3 - 5x^2 + 7x + 9.$$

W poniższej tabeli wpisz wartości w obliczane przez algorytm w linii (*)

	Wartość w
$k = 5$	
$k = 4$	
$k = 3$	
$k = 2$	
$k = 1$	
$k = 0$	

Podaj wynik, jaki zwróci algorytm:

22.4.

Wielomianem parzystym nazywamy wielomian stopnia $2n$ postaci

$$R(x) = a_n x^{2n} + a_{n-1} x^{2n-2} + \dots + a_2 x^4 + a_1 x^2 + a_0,$$

tzn. taki, w którym występują tylko parzyste potęgi zmiennej x .

Bazując na schemacie Hornera, napisz algorytm o poniższej specyfikacji (w pseudokodzie lub wybranym języku programowania), który oblicza wartość parzystego wielomianu $R(x)$.

Dane:

- n — liczba całkowita, $n \geq 0$,
- x — liczba rzeczywista,
- a_0, a_1, \dots, a_n — liczby rzeczywiste.

Wynik: wartość $R(x)$.

Przy ocenie rozwiązania będzie brana pod uwagę liczba operacji mnożenia i dodawania wykonywanych przez algorytm.

Zadanie 23.

Wiązka zadań *Obliczanie całkowitego pierwiastka kwadratowego*

Całkowity pierwiastek kwadratowy z liczby naturalnej x jest największą liczbą naturalną p , która spełnia nierówność $p^2 \leq x$. Poniższy algorytm służy do obliczania tej wartości przybliżonej.

Specyfikacja

Dane:

x — liczba naturalna

Wynik:

p — liczba naturalna spełniająca warunek $p^2 \leq x$ i $(p+1)^2 > x$

Algorytm

```
i ← 0  
a ← 0  
r ← 1  
dopóki a ≤ x wykonuj  
    i ← i + 1  
    a ← a + r  
    r ← r + 2  
zwróć i – 1 i zakończ
```

23.1.

Niech $x = 21$. Przeanalizuj działanie powyższego algorytmu i uzupełnij wartości zmiennych a i r dla kolejnych wartości i podanych w tabeli.

Wartość i	Wartość a	Wartość r
0		
1		
2		
3		
4		
5		

23.2.

Zdecyduj, które z poniższych zdań są w odniesieniu do opisanego algorytmu prawdziwe (P), a które fałszywe (F). **Zaznacz znakiem X** odpowiednią rubrykę w tabeli .

	P	F
Konstruuje ciąg kwadratów kolejnych liczb naturalnych.		
Znajduje dokładną wartość pierwiastka z liczby x .		
Oblicza kolejne nieparzyste liczby naturalne.		
Wykonuje dokładnie tyle iteracji pętli, ile wynosi pierwiastek całkowity z liczby x .		

23.3.

Napisz algorytm obliczania całkowitego pierwiastka z liczby naturalnej, który wykorzystuje następującą zależność rekurencyjną definiującą ciąg x_n zbieżny do pierwiastka kwadratowego z liczby x :

$$\begin{cases} x_0 = \frac{x}{2} \\ x_{n+1} = \frac{1}{2} \left(x_n + \frac{x}{x_n} \right) \end{cases}$$

Specyfikacja

Dane:

x — liczba naturalna

Wynik:

p — liczba naturalna spełniająca warunek $p^2 \leq x$ i $(p+1)^2 > x$

Zadanie 24.

Wiązka zadań Poszukiwania

Dana jest liczba naturalna $n > 0$ oraz uporządkowana tablica liczb całkowitych $T[1..n]$. Rozważmy następującą funkcję F dla trzech argumentów p, k, e , które są liczbami całkowitymi dodatnimi:

funkcja $F(p, k, e)$

jeżeli ($k = p$)

jeżeli ($T[p] > e$)

zwróć p i zakończ

w przeciwnym razie

zwróć $p + 1$ i zakończ

w przeciwnym razie

$s \leftarrow (p+k) \text{ div } 2$

jeżeli $T[s] > e$

zwróć $F(p, s, e)$

w przeciwnym razie

zwróć $F(s+1, k, e)$

24.1.

Przeanalizuj działanie funkcji F i uzupełnij poniższą tabelkę dla $p = 1, k = 5, e = 10$:

T	$F(p, k, e)$
[3, 4, 6, 8, 9]	
[15, 16, 18, 22, 24]	
[2, 10, 16, 24, 26]	
[1, 3, 10, 10, 18]	

24.2.

Zdecyduj, które z dokończeń podanego niżej zdania czynią z niego zdanie prawdziwe. Zaznacz to **znakiem X** w odpowiednich miejscach tabeli.

Funkcja F wykorzystuje

metodę zachłanną.	
strategię „dziel i zwyciężaj”.	
programowanie dynamiczne.	

24.3.

Zdecyduj, które z dokończeń podanego niżej zdania czynią z niego zdanie prawdziwe. Zaznacz to **znakiem X** w odpowiednich miejscach tabeli.

Liczba wywołań rekurencyjnych funkcji F dla początkowych wartości $p = 1$ oraz $k = n$, będącej potągą dwójki, jest w najgorszym przypadku równa

$n \text{ div } 2.$	
$\sqrt{n}.$	
$\log_2 n.$	

24.4.

Napisz algorytm, który korzystając z funkcji F, obliczy, ile jest liczb należących do przedziału $[a, b]$ w uporządkowanej niemalejąco tablicy $T[1..n]$, składającej się z liczb całkowitych dodatnich.

Specyfikacja

Dane:

n — liczba całkowita dodatnia

$T[1..n]$ — uporządkowana tablica n liczb całkowitych, złożona z różnych liczb całkowitych dodatnich, taka że $T[1] \leq T[2] \leq \dots \leq T[n]$

a — liczba całkowita dodatnia

b — liczba całkowita dodatnia

Wynik:

w — liczba elementów tablicy $T[1..n]$ należących do przedziału $[a, b]$.

Zadanie 25.

Wiązka zadań Palindromy

Palindrom to słowo lub zdanie, które czytane od lewej do prawej i od prawej do lewej brzmi tak samo. W zdaniu, które jest palindromem, ignorujemy odstępy między słowami. Ważne jest tylko, że kolejność liter przy czytaniu od lewej do prawej i od prawej do lewej jest taka sama. Dla potrzeb zadania przyjmujemy, że używamy tylko małych liter alfabetu łacińskiego. Poniżej podajemy przykłady słów i zdań, które są palindromami:

kajak

anna

elf układał kufle

trafilis popili fart

ewo zeby tu były buty bezowe

Poniższy algorytm sprawdza, czy podane na wejściu słowo jest palindromem.

Specyfikacja

Dane:

$S[1..d]$ — słowo zapisane w tablicy znaków, gdzie $d > 1$ oznacza liczbę znaków w słowie

Wynik:

TAK — gdy słowo S jest palindromem, NIE — w przeciwnym przypadku

Algorytm:

```
d ← długość(S)
i ← d div 2
(*) dopóki (i > 0) i (S[i] = S[d - i+1]) wykonuj
    i ← i - 1
jeżeli i = 0
    zwróć TAK i zakończ,
w przeciwnym razie
    zwróć NIE i zakończ
```

Uwaga: `div` oznacza operator dzielenia całkowitego, a wartością funkcji `długość` jest liczba znaków w słowie.

25.1.

Podaj przykład słowa o długości 9, niebędącego palindromem, dla którego powyższy algorytm wykonuje największą możliwą liczbę powtórzeń w pętli oznaczonej (*).

25.2.

Napisz algorytm, który sprawdza, czy zdanie jest palindromem.

Specyfikacja

Dane:

Zdanie[1..d] — zdanie o długości d znaków zapisane w tablicy znaków, składające się z małych liter alfabetu łacińskiego i spacji, w tym co najmniej jednej litery.

Wynik:

TAK, gdy *Zdanie* jest palindromem, NIE — w przeciwnym razie

Zadanie 26.

Wiązka zadań Podobieństwo słów

Słowo *X* nazywać będziemy *k-podrzędnym* względem słowa *Y*, jeśli **wszystkie litery występujące w X występują również w Y** oraz w słowie *Y* występuje dokładnie *k różnych liter*, które **nie** występują w słowie *X*.

Słowo *X* jest podrzędne względem słowa *Y*, gdy *X* jest *k*-podrzędne względem *Y* dla jakiegoś $k \geq 0$.

Przykład

Słowo *X=ABCAB* jest 1-podrzędne względem słowa *Y=BAACD* (w słowie *X* występują litery A, B, C; w słowie *Y* występują litery A, B, C, D). Podobnie słowo *X=ABCAB* jest 0-podrzędne względem *BAC* oraz nie jest podrzędne względem słów *ABDAB* i *ABAB* (litera *C* występuje w słowie *X*, a nie występuje w słowach *ABDAB* i *ABAB*). Zamieniając słowa rolami, możemy stwierdzić, że słowo *ABDAB* nie jest podrzędne względem słowa *ABCAB*, a słowo *ABAB* jest 1-podrzędne względem *ABCAB*.

Uwaga

W poniższych zadaniach przyjmujemy, że w słowach mogą występować tylko litery $A, B, C, D, E, F, G, H, I, J$. Ponadto w algorytmach dostępna jest funkcja $dlugosc$, która zwraca długość słowa będącego jej argumentem, oraz funkcja kod o wartościach podanych w poniższej tabeli:

litera	A	B	C	D	E	F	G	H	I	J
$kod(\text{litera})$	1	2	3	4	5	6	7	8	9	10

26.1.

Uzupełnij poniższą tabelę, wpisując w kolumnie Podrzędność słowo NIE, jeśli słowo X nie jest podrzędne względem słowa Y , a w przeciwnym wypadku — liczbę k taką, że X jest k -podrzędne względem Y .

Słowo X	Słowo Y	Podrzędność
$HHGGFFEEDDCCBAA$	$ABCDEFGH$	
$DCBADCBA$	$FGHABCJD$	
$ABCDE$	$ABCCBAE$	
$AAAAA$	AA	
ABA	ACA	
$ACEGJ$	$ABCDEFGHIJ$	

26.2.

Dany jest następujący **algorytm A**:

```
dla i=1,2,...,10 wykonuj
    Czyjest[i] ← fałsz
    d ← dlugosc(Y)
    dla i=1,2,...,d wykonuj
        lit ← Y[i]
        Czyjest[kod(lit)] ← prawda
        d ← dlugosc(X)
        czyp ← prawda
        dla i=1,2,...,d wykonuj
            lit ← X[i]
            czyp ← czyp i Czyjest[kod(lit)]
        jeżeli czyp=prawda
            zwróć 1
        w przeciwnym razie
            zwróć 0
```

Podaj wynik działania powyższego algorytmu dla wartości X i Y z poniższej tabeli:

X	Y	wynik algorytmu A
$HHGGFFEEDDCCBAA$	$ABCDEFGH$	
$DCBADCBA$	$FGHABCJD$	
$ABCDE$	$ABCCBA$	
$AAAAA$	AA	
AA	$AAAAA$	
$ACEGJ$	$ABCDEFGHIJ$	

Uzupełnij podaną poniżej specyfikację algorytmu A.

Specyfikacja

Dane: X, Y — słowa, w których występują tylko litery ze zbioru $\{A, B, C, D, E, F, G, H, I, J\}$

Wynik:
.....

26.3.

Uzupełnij brakujące fragmenty poniższego algorytmu B, tak aby realizował on podaną specyfikację.

Specyfikacja

Dane:

X, Y — słowa, w których występują tylko litery ze zbioru $\{A, B, C, D, E, F, G, H, I, J\}$

Wynik:

- k — liczba całkowita, taka że
 - X jest k -podrzędne względem Y , jeśli $k \geq 0$,
 - X nie jest podrzędne względem Y , gdy $k = -1$.

Algorytm B:

```
dla i=1,2,...,10 wykonuj
    Czy_x[i] ← fałsz
    Czy_y[i] ← fałsz
    dx ← dlugosc(X)
    dla i=1,2,...,dx wykonuj
        lit ← X[i]
        Czy_x[kod(lit)] ← prawda
    dy ← dlugosc(Y)
    dla i=1,2,...,dy wykonuj
        lit ← Y[i]
        Czy_y[kod(lit)] ← prawda
    k ← 0
    dla i=1,2,...,10 wykonuj
        jeśli Czy_y[i]=prawda oraz Czy_x[i]= .....
            k← .....
        jeśli Czy_y[i]=..... oraz Czy_x[i]=prawda
            zwróć -1 i zakończ
    zwróć k
```

26.4.

Słowa X i Y nazywać będziemy *równoważnymi*, jeśli każda litera występuje tyle samo razy w słowie X i w słowie Y .

Przykład

Słowa $ABCA$ i $BCAA$ są równoważne, natomiast $ABCA$ nie jest równoważne ze słowami $ABCC$, $BABB$ i $ABCAD$.

Podaj algorytm, który sprawdza, czy dwa podane słowa są równoważne. Twój algorytm powinien realizować następującą specyfikację:

Specyfikacja

Dane:

X, Y — słowa, w których występują tylko litery ze zbioru $\{A, B, C, D, E, F, G, H, I, J\}$

Wynik:

1 — gdy X i Y są równoważne, 0 — w przeciwnym razie.

Zadanie 27.

Wiązka zadań *Dopasowanie z błędem*

W podanym *tekście*, złożonym z małych liter alfabetu łacińskiego, wyszukujemy słowo zwanego *wzorcem*. Celem jest znalezienie takiego fragmentu tekstu, który jest albo dokładnie równy wzorcowi, albo jest mu równy z jednym błędem, czyli różni się od niego najwyżej jedną literą.

Na przykład wzorzec *para* można znaleźć w tekście *parawan* albo *aparat*, a z jednym błędem — w tekście *opera* albo *spadanie*. Będziemy zakładać, że tekst jest co najmniej tak samo długi jak wzorzec, a wzorzec składa się z co najmniej dwóch liter.

27.1.

Dla podanych wzorców i tekstów podaj, czy wzorzec występuje w tekście dokładnie, z jednym błędem, czy też w ogóle w nim nie występuje. Do tabeli wpisz odpowiednio „dokładnie”, „z błędem” lub „nie”.

Wzorzec	Tekst	W jaki sposób wzorzec występuje w tekście?
<i>para</i>	<i>opera</i>	z błędem
<i>para</i>	<i>aparat</i>	dokładnie
<i>kran</i>	<i>karawana</i>	
<i>sport</i>	<i>bezspornie</i>	
<i>ryt</i>	<i>zakryty</i>	
<i>sofa</i>	<i>solanka</i>	

27.2.

Może się zdarzyć, że wzorzec występuje w tekście więcej niż raz: na przykład w słowie *rabarbar* wzorzec *bar* występuje dwukrotnie. Wystąpienia mogą się częściowo nakładać: wzorzec *issi* występuje dwukrotnie w *mississippi*.

Podaj przykład tekstu o długości 8 oraz wzorca o długości 4, dla których wzorzec występuje w tekście (dokładnie) przynajmniej trzykrotnie.

27.3.

Podaj algorytm (w pseudokodzie lub wybranym języku programowania), który dla danego wzorca i danego tekstu, rozstrzygnie, czy wzorzec występuje w tekście (dokładnie lub z błędem).

Algorytm powinien wypisywać TAK, jeśli wzorzec występuje, NIE — w przeciwnym wypadku.

Dane:

dodatnie liczby całkowite m i n , $n \geq m$

wzorzec $[1..m]$, tekst $[1..n]$, napisy złożone z małych liter alfabetu łacińskiego

Wynik:

słowo „TAK”, jeśli wzorzec występuje w tekście (dokładnie lub z błędem), zaś słowo „NIE”, jeśli nie występuje.

Zadanie 28.

Wiązka zadań *Słabe palindromy*

Niech $W[i,j]$ oznacza część słowa W , zaczynającą się na i -tej literze, a kończącej na j -tej literze W . Na przykład dla słowa $W = ABCDEF$ mamy $W[3,5] = CDE$. Z kolei $W[i]$ oznaczać będzie i -tą literę słowa W , czyli $W[i]=W[i,i]$.

Uwaga: W treści zadania nie będziemy odróżniać pojedynczych znaków od całych napisów, co jest charakterystyczne dla niektórych języków programowania.

Niech W będzie złożonym z liter A i B słowem o długości m . Słowo W nazywamy *słabym A-palindromem*, jeśli spełnia ono jeden z dwóch warunków:

1. $W = „A”$, czyli W jest słowem złożonym jedynie z litery A ;
2. długość słowa m jest liczbą parzystą oraz spełnione są jednocześnie poniższe warunki:
 - a. $W[1] = W[m]$
 - b. $W[1,m/2]$ lub $W[m/2+1,m]$ jest słabym A-palindromem.

Przykłady

1. Jedynym dwuliterowym słabym A-palindromem jest AA .
2. Słowa $W_1=AAAA$ oraz $W_2=AABA$ są słabymi A-palindromami, oba spełniają warunek 2a (gdyż $W_1[1]=W_1[4]=A$ oraz $W_2[1]=W_2[4]=A$). Ponadto $W_1[1,2]=W_2[1,2]=AA$ jest także słabym A-palindromem, co zapewnia spełnienie warunku 2b zarówno dla W_1 jak i W_2 .
3. Słowo $W=AAAAAA$ nie jest słabym A-palindromem, ponieważ nie jest spełniony warunek 2b. Słowo W ma długość $m=6$, co oznacza, że słowa $W[1,m/2]$ oraz $W[m/2+1,m]$ mają długość nieparzystą równą 3. W rezultacie żadne ze słów: $W[1,m/2]=AAA$, $W[m/2+1,m]=AAA$ nie jest słabym A-palindromem.
4. Słowo $AAABBAAA$ nie jest słabym A-palindromem, natomiast $AABABBAA$ jest słabym A-palindromem.

28.1.

Uzupełnij poniższą tabelę, ustalając, które słowa spełniają warunki 2a i 2b. oraz które są słabymi A-palindromami. W przypadku słabych A-palindromów w kolumnie *Uzasadnienie* wskaz tą połowę słowa, która zapewnia spełnienie warunku 2b (jeśli obie połówki zapewniają spełnienie 2b, to wystarczy wskazać tylko jedną). Dla słów, które nie są słabymi A-palindromami, kolumnę *Uzasadnienie* zostaw pustą.

Słowo	Czy jest spełniony warunek		Czy słowo jest słabym A-palindromem?	Uzasadnienie
	2a?	2b?		
AABAABAA	tak	tak	tak	AABA
AAABBAAA	tak	nie	nie	
AAABBAAB	nie	nie	nie	
AAAABBAA				
ABBBABAA				
ABAAAAABA				
AAAAAAAAAA				

28.2.

Rozważmy mające różną długość słabe A-palindromy, w których występuje najmniej liter A . Wiemy, że jest tylko jeden słaby A-palindrom jednoliterowy równy A oraz jeden słaby A-palindrom dwuliterowy równy AA . Nietrudno sprawdzić, że wszystkie czteroliterowe słabe A-palindromy to: $AAAA$, $ABAA$, $AABA$. Zatem najmniejsza liczba liter A w słabym A-palindromie o długości 1 jest równa 1, najmniejsza liczba liter A w słabym A-palindromie o długości 2 jest równa 2, a najmniejsza liczba liter A w słabym A-palindromie o długości 4 jest równa 3.

Uzupełnij puste pola w poniższym zestawieniu:

m	najmniejsza liczba liter A słabego A-palindromu o długości m	słaby A-palindrom o długości m i najmniejszej liczbie liter A
2	2	AA
4	3	
8		

Uzupełnij puste pola w poniższej tabeli, podając najmniejszą liczbę liter A w słabych A-palindromach o podanych poniżej długościach, uzupełniając puste pola w poniższej tabeli.

m	najmniejsza liczba liter A słabego A-palindromu o długości m
16	
32	
2^{10}	
2^{20}	

28.3.

Podaj algorytm sprawdzający, czy podane na wejściu słowo W jest słabym A-palindromem. Twój algorytm powinien działać zgodnie z następującą specyfikacją:

Specyfikacja

Dane:

słowo W

Wynik:

„Tak”, gdy w jest słabym A-palindromem,

„Nie”, gdy w nie jest słabym A-palindromem.

Zadanie 29.

Wiązka zadań *Alfabet kulkowy*

W Kulkolandii do zapisu wiadomości używa się kulek o dwóch kolorach, czarnym i białym. Każda litera alfabetu jest zapisywana za pomocą pewnej liczby kulek odpowiednio uporządkowanych tak, jak to przedstawiono w poniższej tabeli. Zauważ, że zapis każdej litery kończy się dwiema czarnymi kulkami:

A	●●	I	○○●○●●	R	●○○○○●●
B	○●●	J	○●○○●●	S	●○○●○●●
C	○○●●	K	●○○○●●	T	●○●○○●●
D	●○●●	L	●○●○●●	U	○○○○○○●●
E	○○○●●	M	○○○○○●●	W	○○○○●○●●
F	○●○●●	N	○○○●○●●	X	○○○●○○●●
G	●○○●●	O	○○●○○●●	Y	○○●○○○●●
H	○○○○●●	P	○●○○○●●	Z	○●○○○○●●

29.1.

Odczytaj poniższą wiadomość zapisaną alfabetem kulkowym:

○○○○○●●●●●●○●○○●●○○○○○●●●●●○○○○●●●●●

29.2.

Do dyspozycji masz trzy operacje:

- *pusty(ciag)*, która inicjuje pusty ciąg kulek;
- *pobierz(ciag)*, która usuwa pierwszą kulę z niepustego ciągu kulek i podaje ją jako wynik;
- *dolacz(ciag, kulka)*, która do podanego ciągu dołącza na końcu wskazaną kulę.

Zapisz algorytm (w pseudokodzie lub w wybranym języku programowania), który z danego ciągu kulek *wiadomosc* usuwa początkowy fragment odpowiadający pierwszej literze i zapisuje go jako ciąg kulek w zmiennej *litera*.

Specyfikacja

Dane:

wiadomosc — ciąg kulek z zakodowanym napisem

Wynik:

litera — ciąg kulek odpowiadający pierwszej literze z zakodowanego napisu

wiadomosc — ciąg kulek pozostałych po pobraniu kulek odpowiadających pierwszej literze

Uwaga: połączenie wynikowych *litera* i *wiadomosc* daje w rezultacie wejściową *wiadomosc*

29.3.

Do dyspozycji masz operacje:

- *pusty(tekst)*, która inicjuje pusty tekst;
- *pobierz_litere(ciag)*, która usuwa z niepustego ciągu kulkowego z zapisaną wiadomością kulki odpowiadające pierwszej literze, a jako wynik podaje literę

- odpowiadającą usuniętym kulkom;
- *dopisz(tekst, litera)*, która do podanego tekstu (może on być pusty) dołącza na końcu wskazaną literę;
- *czy_sa_kulki(ciag)*, podającą wartość PRAWDA, gdy w podanym ciągu znajduje się przynajmniej jedna kulka, a wartość FAŁSZ, gdy w podanym ciągu nie ma żadnej kulki.

Zapisz algorytm (w pseudokodzie lub w wybranym języku programowania), który dekoduje napis zapisany kulkami.

Specyfikacja

Dane:

ciag — ciąg kulek z zakodowaną wiadomością

Wynik:

wiadomosc — tekst z dekodowaną wiadomością

Zadanie 30.

Wiązka zadań Szyfrowanie

Szyfrowanie z kluczem k (k — liczba całkowita większa lub równa 0) polega na zastąpieniu każdego znaku wiadomości jawnej innym znakiem o pozycji przesuniętej w alfabetie o k znaków względem znaku szyfrowanego. Przy szyfrowaniu znaku należy postępować w sposób cykliczny, tzn. po osiągnięciu końcowego znaku alfabetu należy powrócić na jego początek. Rozważamy tylko wielkie litery alfabetu angielskiego, tj. litery o kodach dziesiętnych ASCII od 65 (dla znaku A) do 90 (dla znaku Z).

30.1.

Mając do dyspozycji funkcję *kod(x)*, która dla danego znaku x podaje dziesiętny kod ASCII tego znaku (np. *kod(A) = 65*), oraz funkcję *znak(y)*, która dla danego dziesiętnego kodu ASCII podaje znak odpowiadający temu kodowi (np. *znak(77) = M*), napisz funkcję *szyfruj(zn,k)*, szyfrującą znak zn szyfrem z kluczem k .

Przykład działania funkcji: *szyfruj(M,327) = B*.

30.2.

Aby utrudnić proces deszyfrowania, postanowiono dla każdego znaku w słowie zastosować inny klucz. Kluczem znaku w słowie będzie numer jego pozycji w tym słowie.

Przykład: w słowie BIT, dla znaku B mamy $k=1$, dla I — $k=2$, a dla T — $k=3$. Zaszyfrowane słowo BIT to CKW.

Uzupełnij poniższą tabelę. Zaszyfruj opisanym sposobem podane słowo jawne oraz odszyfruj słowo zaszyfrowane.

Słowo jawne	Słowo zaszyfrowane
INFORMATYKA	
	LQPTZZLZ

30.3.

Dane jest słowo S zaszyfrowane sposobem opisanym w zadaniu 2. Podaj algorytm (w postaci pseudokodu lub kodu wybranego języka programowania) deszyfrujący słowo S .

Dane:

- n — liczba znaków w słowie S ($n > 0$)
- $S[1..n]$ — zaszyfrowane słowo

Wynik:

- $J[1..n]$ — słowo jawne, które po zaszyfrowaniu daje słowo S

Zadanie 31.

Wiązka zadań Szyfr skokowy

Szyfrem skokowym z kluczem $k > 0$ nazywać będziemy kodowanie tekstu opisane przez poniższy algorytm, w którym danymi na wejściu są: k — liczba całkowita dodatnia oraz W — tekst.

Algorytm:

```
n ← dlugosc(W)
m ← n div k
jeżeli n mod k ≠ 0
    m ← m + 1
dla i=1,2,...,m wykonuj
    j ← i
    dopóki j ≤ n wykonuj
        wypisz W[j]
        j ← j + m
```

Uwaga: Przyjmujemy, że:

- dostępna jest funkcja $dlugosc$, która zwraca długość słowa będącego jej argumentem;
- kolejne znaki słowa W o długości n oznaczamy przez $W[1], W[2], \dots, W[n]$.

Przykład

Dla $k=3$ i tekstu SZYFROWANIE algorytm wypisze tekst SRNZOIYWEFA.

31.1.

Dla poniższych wartości k i W podaj tekst wypisany przez algorytm:

- $k=3, W=ZADANIE1JESTŁATWE$

Tekst wypisany przez algorytm:.....

- $k=4, W=ZADANIE1JESTPROSTE$

Tekst wypisany przez algorytm:.....

31.2.

Podaj teksty W , dla których powyższy algorytm wypisze na wyjściu podane wartości:

- $k=3, W=.....$

Tekst wypisany przez algorytm: UDOMEWIKAEÓCMD

- $k=4, W= \dots\dots\dots\dots$

Tekst wypisany przez algorytm: DRJTOZEBES

31.3.

Podaj algorytm deszyfrowania tekstu zakodowanego szyfrem skokowym zgodny z poniższą specyfikacją:

Specyfikacja

Dane:

k — liczba całkowita dodatnia

X — tekst

Wynik:

W — tekst, którego szyfr skokowy z kluczem k jest równy W .

Przykład

Dla $k=3$ i $X=SRNZOIYWEFA$ algorytm powinien zwrócić na wyjściu tekst $W=SZYFROWANIE$.

31.4.

Szyfrem mieszanym z kluczem $k>0$ nazywać będziemy następujący sposób kodowania tekstu W o długości n :

1. Dzielimy tekst W na k części tak, że wszystkie części poza ostatnią mają tę samą długość m równą zaokrągleniu w górę liczby n / k . Ostatnia część ma długość mniejszą lub równą m .
2. Tak uzyskane części wpisujemy w kolejnych wierszach prostokątnej tabeli o k wierszach i m liczbie kolumn.
3. Wypisujemy kolejne kolumny, zaczynając od pierwszej, zgodnie z następującą zasadą: kolumny o nieparzystym numerze wypisujemy z góry na dół, a kolumny o parzystym numerze z dołu do góry.

Przykład

Dla $k=3$ i tekstu $W=SZYFROWANIE$ uzyskamy tabelę:

S	Z	Y	F
R	O	W	A
N	I	E	

a zaszyfrowana tekst ma postać SRNIOZYWEAF.

Podaj algorytm szyfrowania tekstu zakodowanego szyfrem mieszanym zgodny z poniższą specyfikacją:

Specyfikacja

Dane:

k — liczba całkowita dodatnia

W — tekst

Wynik:

X — szyfr mieszany tekstu W z kluczem k .

Zadanie 32.

Wiązka zadań *Kompresja*

Dany jest napis złożony z małych liter alfabetu angielskiego, który kompresujemy w następujący sposób: jeżeli w napisie występują kolejno po sobie dwa identyczne ciągi kolejnych znaków, na przykład *piripiri*, to możemy zastąpić je jednokrotnym wystąpieniem tego ciągu, ujętym w nawiasy okrągłe, np. *(piri)*.

Na przykład *xyzxyz* można zapisać jako *(xyz)*, zaś *rabarbar* jako *ra(bar)*.

32.1.

Niektóre fragmenty napisów znajdujących się w tabelce zostały skompresowane podaną powyżej metodą. Uzupełnij tabelkę, odtwarzając oryginalne napisy.

Napis skompresowany	Napis oryginalny
a(cd)a	acdcda
(pur)owy	
(z)(zz)	
(ab)a(abcd)	

32.2.

Podaj algorytm (w pseudokodzie lub wybranym języku programowania), który mając zapisany w tablicy pewien napis, sprawdzi czy jest on (w całości) dwukrotnym powtórzeniem pewnego fragmentu, a jeśli tak, wypisze skompresowany napis w postaci (*fragment*). Jeśli napis wejściowy nie jest powtórzeniem, na wyjście nie należy nic wypisywać.

Dane:

- liczba całkowita $n > 0$ oraz tablica $napis[1..n]$, zawierająca napis złożony z małych liter alfabetu angielskiego

Wynik:

- jeżeli napis wejściowy jest dwukrotnym powtórzeniem tego samego fragmentu, wynikiem są kolejne znaki napisu skompresowanego.

32.3.

Dany jest napis, w którym niektóre fragmenty zostały skompresowane podaną powyżej metodą. Podaj algorytm (w pseudokodzie lub wybranym języku programowania), który mając na wejściu skompresowany napis zapisany w tablicy, wypisze na wyjście kolejne litery napisu oryginalnego.

Dane:

dodatnia liczba całkowita n oraz tablica $napis[1..n]$, zawierająca pewien napis skompresowany metodą opisaną wcześniej.

Wynik:

kolejne znaki oryginalnego napisu, którego skompresowana wersja jest w tablicy $napis$.

Uwaga: W napisie skompresowanym nie ma zagnieżdżeń nawiasów, czyli wewnątrz pary nawiasów nie wystąpią inne nawiasy.

Zadanie 33.

Wiązka zadań Wędrowka po planszy

Wędrowiec podrózuje po kwadratowej planszy rozmiaru $n \times n$. Swoją wędrówkę rozpoczyna na dowolnym polu **pierwszej kolumny** planszy, a na koniec powinien dotrzeć do **ostatniej kolumny**. Będąc w kolumnie $j < n$, wędrowiec może w jednym ruchu przenieść się do kolumny $j+1$ (nie może przenieść się do żadnej innej kolumny). Wartość każdego pola planszy jest liczbą całkowitą. Wartość pola w i -tym wierszu i j -tej kolumnie na planszy A oznaczać będziemy przez $A[i, j]$.

Rozważamy 3 typy wędrowca:

- skaczący, który z pola w kolumnie $j < n$ może przeskoczyć na dowolne pole w kolumnie $j+1$;
- spadający, który z pola $A[i, j]$ może przenieść się tylko na pola $A[k, j+1]$ takie, że $i \leq k \leq n$;
- chodzący, który z pola $A[i, j]$ może przeskoczyć tylko na pola $A[k, j+1]$ takie, że $k=i$ lub $k=i-1$ lub $k=i+1$ oraz $1 \leq k \leq n$.

Przykład

Rozważmy planszę rozmiaru 10×10 . Jeżeli bieżącą pozycję wędrowca jest pole $A[3,4]$, to

- **wędrowiec skaczący** może w jednym ruchu przenieść się do pól $A[1,5], A[2,5], \dots, A[10,5]$,
- **wędrowiec spadający** może przenieść się do pól $A[3,5], A[4,5], \dots, A[10,5]$,
- **wędrowiec chodzący** może przenieść się do pól $A[2,5], A[3,5], A[4,5]$.

Rozważmy następujący algorytm, opisujący trasę jednego z typów wędrowców, dla poniższych danych:

Dane: n — liczba naturalna większa od 1;

A — tablica rozmiaru $n \times n$ wypełniona liczbami całkowitymi.

Uwaga: W poniższym algorytmie $B[0..n+1, 1..n]$ jest tablicą, której wiersze mają numery $0, 1, \dots, n, n+1$, a kolumny $1, 2, \dots, n-1, n$.

Algorytm:

```
dla i=1,2,...,n wykonuj
    jeżeli A[i,1]>0
        B[i,1] ← 1
    w przeciwnym razie
        B[i,1] ← 0
dla j=2,3,...,n wykonuj
    B[0,j] ← 0
    B[n+1,j] ← 0
    dla i=1,2,...,n wykonuj
        jeżeli A[i,j] ≤ 0
            B[i,j] ← 0
        w przeciwnym razie
            jeżeli B[i-1,j-1]=1 lub B[i,j-1]=1 lub B[i+1,j-1]=1
                B[i,j] ← 1
            w przeciwnym razie
                B[i,j] ← 0
d ← 0
dla i=1,2,...,n wykonuj
    jeżeli B[i,n]=1
        d ← 1
zwróć d
```

33.1.

Rozważmy działanie algorytmu dla $n=5$ oraz następującej zawartości tablicy A:

	1	2	3	4	5
1	-2	-1	4	7	8
2	-3	2	3	-10	-2
3	1	-4	-1	5	-5
4	-2	-1	-2	-3	9
5	-1	-5	1	-4	1

Podaj końcową zawartość kolumn 1,2,...,5 tablicy B oraz wartość zwracaną przez algorytm_1 dla powyższych danych.

Tablica B:

	1	2	3	4	5
0					
1					
2					
3					
4					
5					
6					

Wartość zwracana przez algorytm:

33.2.

Uzupełnij specyfikację podanego powyżej algorytmu.

Specyfikacja

Dane:

n — liczba naturalna większa niż 1

A — plansza rozmiaru $n \times n$ wypełniona liczbami całkowitymi.

Wynik:

1 — jeśli istnieje trasa wędrowca typu

i prowadząca tylko przez pola o wartościach

0 — w przeciwnym przypadku

33.3.

Wartością trasy wędrowca nazywamy sumę liczb zapisanych na polach planszy, które wędrowiec odwiedza w trakcie trasy. Podaj algorytm zgodny z poniższą specyfikacją.

Specyfikacja

Dane:

n — liczba naturalna większa niż 1

A — plansza rozmiaru $n \times n$ wypełniona liczbami całkowitymi.

Wynik:

Największa wartość trasy wędrowca typu skaczącego zaczynającej się w pierwszej kolumnie i kończącej się w ostatniej kolumnie.

Przykład

Dla $n=5$ oraz zawartości planszy podanej w zadaniu 1 algorytm powinien zwrócić wartość 23; trasa wędrowca o największej wartości prowadzi przez pola $A[3,1]$, $A[2,2]$, $A[1,3]$, $A[1,4]$ i $A[4,5]$.

33.4.

Podaj algorytm zgodny z poniższą specyfikacją.

Specyfikacja

Dane:

n — liczba naturalna większa niż 1

A — plansza rozmiaru $n \times n$ wypełniona liczbami całkowitymi.

Wynik:

1 — jeśli istnieje trasa wędrowca spadającego zaczynająca się w pierwszej kolumnie, kończąca się w ostatniej kolumnie i przechodząca wyłącznie przez pola o wartościach nieujemnych;

0 — w przeciwnym przypadku.

Przykład

Dla $n=5$ oraz zawartości planszy podanej w zadaniu 1 algorytm powinien zwrócić wartość 0, gdyż trasa spełniająca podane warunki musi zaczynać się w polu $A[3,1]$, z którego można przejść tylko do pól ujemnych w drugiej kolumnie. Dla $n=5$ i poniższej zawartości planszy algorytm powinien zwrócić wartość 1; trasa prowadząca tylko przez pola dodatnie może przechodzić np. przez pola: $A[1,1], A[2,2], A[2,3], A[3,4]$ i $A[4,5]$.

	1	2	3	4	5
1	2	-1	4	7	8
2	-3	2	3	-10	-2
3	1	-4	-1	5	-5
4	-2	-1	-2	3	9
5	-1	-5	-6	-4	1

1.3. Praktyka w teorii

Zadanie 34.

Bajtek i Bituś poznali starożytny chiński sposób testowania pierwszości liczby naturalnej n :

liczba n musi spełniać równanie

$$2^n \bmod n = 2,$$

gdzie \bmod oznacza operator dzielenia modulo, czyli resztę z dzielenia całkowitego.

Uwaga: Starożytny chiński sposób bywa zawodny, istnieją liczby, które spełniają to równanie, a nie są pierwsze; natomiast jeśli równanie nie zachodzi, to n jest na pewno złożona.

Chłopcy postanowili przetestować chińską metodą kolejne liczby $n \geq 2$, przeprowadzając obliczenia tylko na takich liczbach, które można reprezentować bez znaku na 8 bitach. co znaczy, że na każdym etapie obliczeń stosowali tylko liczby całkowite z zakresu od 0 do 255.

Bajtek obliczał potęgi 2^n dla kolejnych wartości n i dla obliczonej wartości wyznaczał resztę z dzielenia przez n .

Bituś skorzystał z praw arytmetyki modularnej. W każdym z kolejnych kroków obliczania potęgi, wynik iloczynu brał modulo n . Na przykład dla $n=3$ obliczenia wykonywał następująco:

$$2^3 \bmod 3 = ((2*2) \bmod 3)*2 \bmod 3 = 1*2 \bmod 3 = 2.$$

Który z nich mógł przetestować pierwszość liczb w większym zakresie?

Podaj możliwie największą wartość n , dla której możliwe było przeprowadzenie testu pierwszości metodą każdego z chłopców.

W obliczeniach Bajtka $n_{max} = \dots$

W obliczeniach Bitusia $n_{max} = \dots$

Komentarz do zadania

Największą liczbą, jaką można zapisać na 8 bitach, jest 255. Jest to największa wartość, jaka może wystąpić w obliczeniach cząstkowych i wynikowych bez ryzyka błędów obliczeń.

W obliczeniach Bajtka największą liczbą będzie 2^n . Aby nie przekroczyła ona wartości 255, n nie może być większe od 7: $2^7=128$ mieści się w 8 bitach, ale $2^8=256$ już się nie zmieści.

W obliczeniach Bitusia n nie może być większe niż 128, ponieważ wówczas

$2^7 = 128 = 128 \bmod n$, a $2*128 = 256$, czego nie można zapisać na 8 bitach.

Zauważ, że mądry wybór algorytmu stwarza, mimo ograniczeń sprzętowych komputera, możliwość realizacji obliczeń w znacznie większym zakresie wartości niż metoda naiwnie prostsza.

Rozwiążanie

W obliczeniach Bajtka $n_{max} = 7$.

W obliczeniach Bitusia $n_{max} = 128$.

Zadanie 35.

Jeden ze sposobów zapisania niezerowej liczby rzeczywistej x w pamięci komputera polega na:

- 1) przedstawieniu tej liczby w postaci iloczynu trzech liczb:

$(-1)^S$ — przy czym S jest równe 0 lub 1 i nazywa się *znakiem*,

M — *mantysy*, która jest liczbą z przedziału $[1;2)$,

2^C — przy czym C jest liczbą całkowitą zwaną *cechą*,

$$x = (-1)^S * M * 2^C.$$

- 2) zapisaniu w pamięci oddziennie: znaku, mantysy i cechy.

Przykład

$$5.5 = 2.75*2^1 = \frac{1.375*2^{\textcircled{2}}}{\text{mantysa}}$$

Przyjmijmy, że każdą liczbę rzeczywistą zapisujemy na **8 bitach**.

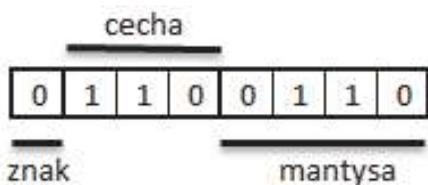
Najstarszy bit (pierwszy z lewej) jest **bitem znaku**: 0 oznacza liczbę dodatnią, 1 — ujemną.

Następne 3 bity reprezentują **cechę** — liczbę całkowitą z zakresu od -4 do 3, reprezentowaną przez trzy cyfry binarne następująco:

-4	000	0	100
-3	001	1	101
-2	010	2	110
-1	011	3	111

Do zapisu **mantisy** pozostają ostatnie 4 bity. Część całkowita mantisy będzie zawsze równa 1, więc wystarczy zapisać jej część ułamkową. Ułamek przedstawiamy w postaci binarnej i zapisujemy cztery pierwsze cyfry rozwinięcia, o wagach: 2^{-1} , 2^{-2} , 2^{-3} i 2^{-4} .

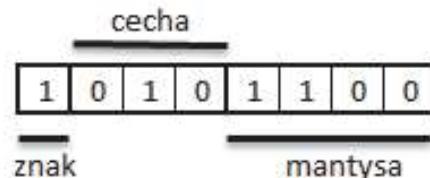
Przykład



Bit znaku ma wartość 0, więc liczba jest dodatnia
 $C = 110_2 - 4 = 6 - 4 = 2$
 $M = 1.0110_2 = 1 + 0.25 + 0.125 = 1.375$
 $x = M \cdot 2^C = 1.375 \cdot 2^2 = 5.5$

35a.

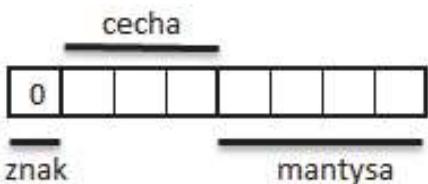
Oceń, czy podane poniżej zdania dotyczące liczby rzeczywistej zapisanej obok w postaci binarnej są prawdziwe, czy fałszywe, stawiając znak X w odpowiedniej kolumnie poniższej tabeli:



		P	F
A	Liczba jest dodatnia.		
B	Cecha ma wartość dziesiętną równą -2.		
C	Mantysa ma wartość dziesiętną równą 0.75.		
D	Liczba ma wartość dziesiętną równą -0.4375.		

35b.

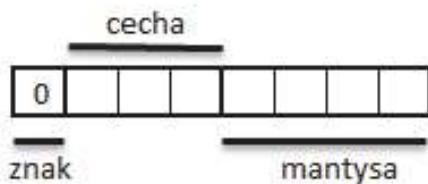
Jaką największą liczbę rzeczywistą x można zapisać, wykorzystując 8 bajtów w sposób opisany wyżej? Wypełnij bity reprezentacji binarnej tej liczby i podaj jej wartość dziesiętną.



$$C = \dots$$
$$M = \dots$$
$$x = \dots$$

35c.

Jaką najmniejszą liczbę nieujemną można zapisać w opisanej wyżej reprezentacji 8-bitowej? Wypełnij bity jej reprezentacji binarnej i podaj jej wartość dziesiętną.



$$C = \dots$$

$$M = \dots$$

$$x = \dots$$

Komentarz do zadania

Zauważ, że w przyjętej reprezentacji:

- można zapisać jedynie wybrane liczby z przedziału od -15.5 do 15.5,
- nie da się zapisać liczby dokładnie równej zero,
- kolejne dwie liczby mogą różnić się od siebie nie mniej niż o 0.125.

W praktyce liczby rzeczywiste zapisywane są na większej liczbie bitów: 32 (w pojedynczej precyzyji) lub 64 (w podwójnej precyzyji). Szerokości cechy i mantysy są więc dużo większe. Nie zmienia to jednak istoty problemu: zakres wartości liczbowych możliwych do zapisania jest ograniczony, a wielu wartości należących do tego zakresu nie da się dokładnie zapisać.

Zadanie 36.

Dane są tabele Uczniowie i Oceny

Uczniowie:

Id_u	Imie	Nazwisko
1	Marek	Wolski
2	Hanna	Wieczorek
3	Irena	Iwanicka
4	Wojciech	Moscicki
5	Janina	Idzik
6	Arkadiusz	Czarnecki

Oceny

Id_u	Wynik
1	2
1	3
2	5
2	4
3	5
4	2
4	5
5	2
6	4
6	3

```
SELECT Uczniowie.Nazwisko  
FROM Uczniowie  
JOIN Oceny ON Uczniowie.Id_u = Oceny.Id_u  
GROUP BY Uczniowie.Nazwisko  
HAVING Avg(Oceny.Wynik) >= 4.5;
```

Zaznacz nazwiska, które pojawią się w wyniku powyższego zapytania.

	T	N
Wolski		
Wieczorek		
Iwanicka		
Moscicki		
Idzik		
Czarnecki		

Zadanie 37.

Dana jest tabela Produkty:

Id	Nazwa	Cena
1	banany	3,20
2	jabłka	2,60
3	winogrona	7,20
4	gruszki	4,10
5	orzechy	5,30
6	pomarańcze	4,80
7	maliny	6,60
8	śliwki	4,10

```
SELECT Nazwa  
FROM Produkty  
ORDER BY Cena DESC;
```

Wśród pierwszych trzech wierszy wyniku dla powyższego zapytania pojawią się:

	T	N
gruszki		
orzechy		
pomarańcze		
maliny		
śliwki		

Zadanie 38.

Dana jest tabela Agenci:

ID_agenta	Obszar_dzialania
A001	Katowice
A002	Katowice
A003	Warszawa
A004	Warszawa
A005	Kraków
A006	Kraków
A007	Katowice

```
SELECT obszar_dzialania, COUNT(*)  
FROM Agenci  
GROUP BY obszar_dzialania;
```

Wynik powyższego zapytania to:

	T	N
Katowice 1 Warszawa 2 Kraków 3		
Katowice 3 Kraków 2 Warszawa 2		
Katowice A001 Kraków A005 Warszawa A003		
A00* 3		

Zadanie 39.

Wiązka zadań *Praktyka w teorii — Grafika*

Na stronie internetowej aquaparku postanowiono umieścić zdjęcia zachęcające do odwiedzin tego miejsca. Zarządcy aquaparku zależy, aby ze strony internetowej mogły korzystać także osoby mające jedynie dostęp do łącza internetowego niskiej prędkości (np. 1 Mbit/s). Jedno ze zdjęć o nazwie aquapark1 zostało przekształcone za pomocą programu graficznego i zapisane jako aquapark2. W tabeli przedstawiono oba zdjęcia wraz z ich parametrami.

	aquapark1
typ obrazu:	jpeg
data:	14.08.2014
rozmiar:	3072 x 2304
głębia w bitach:	24bits
rozmiar pliku:	1,17 MB
	aquapark2
typ obrazu:	jpeg
data:	14.08.2014
rozmiar:	1024 x 768
głębia w bitach:	8bits
rozmiar pliku:	195 KB

39.1.

Który obraz powinien zostać dodany do galerii zdjęć na stronie aquaparku? Uzasadnij swój wybór.

39.2.

Dział promocji aquaparku zamierza przygotować do druku folder, w którym umieści różne zdjęcia. Wybierz z poniższej listy format pliku graficznego, który zapewni najlepszą jakość druku. Uzasadnij swój wybór.

TIFF	JPEG	GIF
------	------	-----

39.3.

Ile pamięci zajmie bitmapa 1024 x 768 pikseli, jeśli zapisano ją w systemie RGB, przeznaczając na każdą składową 8 bitów? Wynik podaj w kilobajtach.

Zadanie 40.

Wiązka zadań *Praktyka w teorii*

Dla następujących zdań **zaznacz znakiem X**, która odpowiedź jest prawdziwa (P), a która jest fałszywa (F).

40.

W kolumnach A, B, C, D arkusza kalkulacyjnego w wierszach od 1 do 200 są wpisane oceny uczniów w postaci liczb całkowitych z przedziału <1,6> z następujących przedmiotów: język polski, język angielski, geografia i biologia.

Przykład

	A	B	C	D	E	F	G	H	I
1	1	4	5	6					
2	3	3	1	5					
3	2	5	2	2					
4	5	2	5	3					
.....									
200	6	2	5	3					

Aby w poszczególnych kolumnach A, B, C, D obliczyć liczbę ocen bardzo dobrych (o wartości liczbowej 5) i umieścić wyniki w komórkach G1 : J1, można w komórce G1 zastosować następującą formułę i skopiować ją do pozostałych komórek:

P	F
LICZ.JEŻELI (B\$1:B\$200; "=5") .	
LICZ.JEŻELI (A\$1:A\$200; "=5") .	
LICZ.JEŻELI (\$A\$1:\$A\$200; "=5") .	
LICZ.JEŻELI (A1:A200; "=5") .	

41.

Numer PESEL ma 11 cyfr. Dla osób urodzonych w latach 2000-2009 pierwszą cyfrą numeru PESEL jest zero. Aby dane w pliku tekstowym, zawierające kolumnę z numerem PESEL, zostały poprawnie zaimportowane do arkusza kalkulacyjnego (jako ciągi jedenastu znaków), należy podczas importu nadać tej kolumnie format:

P	F
tekstowy.	
liczbowy.	
ogólny, a po zaimportowaniu zmienić go na format tekstowy.	
ogólny, a po zaimportowaniu zmienić go na specjalny format o nazwie „Numer PESEL”.	

Zadanie 42.

Wiązka zadań Podział tablicy

Rozważamy następujący algorytm.

Dane:

tablica liczb naturalnych $T[1..n]$

Algorytm:

```
x ← T[1]
i ← 0
j ← n+1
wykonuj
    wykonuj
        j ← j-1
    (*)   dopóki T[j] > x
        wykonuj
            i ← i+1
    (**)
        dopóki T[i] < x
        jeżeli i < j
    (***)   zamień(T[i], T[j])
        w przeciwnym razie
            zakończ
```

Uwaga: funkcja *zamień*(T[i], T[j]) zamienia miejscami wartości T[i] oraz T[j].

42.1.

Przeanalizuj działanie algorytmu i podaj łączną liczbę operacji porównania, jakie zostaną wykonane w wierszach oznaczonych (*) i (**) dla danych zapisanych w poniższej tabeli:

Tablica T	Liczba operacji porównania wykonanych w wierszu oznaczonym (*)	Liczba operacji porównania wykonanych w wierszu oznaczonym (**)
4, 2, 5, 8, 1, 9, 7, 6, 3		
5, 4, 3, 2, 1, 6, 7, 8, 9, 10		
1, 2, 3, ..., 100		
100, 99, 98, ..., 1		

42.2.

Przeanalizuj działanie algorytmu i podaj łączną liczbę operacji zamiany, jakie zostaną wykonane w wierszu oznaczonym (***) dla danych zapisanych w poniższej tabeli:

Tablica T	Liczba operacji zamiany wykonanych w wierszu oznaczonym (***)
4, 2, 5, 8, 1, 9, 7, 6, 3	
5, 4, 3, 2, 1, 6, 7, 8, 9, 10	
1, 2, 3, ..., 100	
100, 99, 98, ..., 1	

Zadanie 43.

Wiązka zadań *Smartfon*

Rozważmy następujące aplikacje:

- serwis pogodowy,
- katalog książek biblioteki szkolnej,
- elektroniczny dziennik lekcyjny,
- edytor tekstu,
- serwis informacji turystycznej Wrocławia,
- wyszukiwanie optymalnej trasy samochodowej,
- arkusz kalkulacyjny,
- kompilator języka programowania,
- system komputerowego składu tekstu.

43.1.

Wskaż wśród powyższych aplikacji dwie, dla których przydatna jest informacja o położeniu geograficznym użytkowników korzystających z urządzeń mobilnych (np. smartfonów). Dla każdej z wybranych aplikacji opisz w jednym lub dwóch zdaniach sposób wykorzystania danych o lokalizacji użytkowników.

43.2.

Wskaż wśród powyższych aplikacji co najmniej dwie, których przydatność jest istotnie ograniczona w sytuacji, gdy użytkownik ma jedynie dostęp do urządzenia mobilnego (np. smartfon, tablet). Dla każdej z wybranych aplikacji uzasadnij swój wybór w co najwyżej dwóch zdaniach.

43.3.

Zdjęcia wykonywane na smartfonie Franka gromadzone są na jego koncie w chmurze obliczeniowej, o ile smartfon jest podłączony do Internetu. W przypadku braku dostępu do Internetu zdjęcia gromadzone są w pamięci podręcznej. Przy obecnych ustawieniach smartfonu często występuje problem braku miejsca w pamięci podręcznej, uniemożliwiający robienie dużej liczby zdjęć. Franek chciałby zmienić ustawienia tak, aby możliwe było zachowanie w pamięci większej liczby zdjęć.

Wśród podanych niżej sposobów wskaż te, które mogą pomóc w rozwiązyaniu problemu Franek (zaznacz znakiem X która odpowiedź jest prawdziwa, a która jest fałszywa):

	P	F
zmiana formatu zapisu zdjęć na mapę bitową		
zmniejszenie rozdzielczości zapisywanych zdjęć		
zastosowanie kodów korekcji CRC, opartych na bitach parzystości		
obniżenie jakości kompresji zdjęć (w formacie JPEG)		

43.4.

Franek założył konto w banku i zamierza korzystać z bankowości internetowej. Wśród poniższych funkcjonalności wskaż te, które mogą służyć zabezpieczaniu usług bankowości internetowej przed nieuprawnionym dostępem (zaznacz znakiem X która odpowiedź jest prawdziwa, a która jest fałszywa):

	P	F
protokół transferu plików FTP (File Transfer Protocol).		
protokół SSL (Secure Socket Layer).		
uwierzytelnianie użytkownika przy pomocy hasła lub PIN.		
kompresja dysku twardego.		
hasła jednorazowe generowane przez układy kryptograficzne i dostarczane kanałami informacyjnymi alternatywnymi dla Internetu.		

1.4. Test z ogólnej wiedzy informatycznej

Zadanie 44.

Zdecyduj, które z dokończeń podanego niżej zdania czynią z niego zdanie prawdziwe (P), a które fałszywe (F). Zaznacz to **znakiem X** w odpowiednich miejscach tabeli.

Liczba 100110010_2

	P	F
jest dwa razy większa od liczby 10011001_2 .		
jest dwa razy mniejsza od liczby 1001100100_2 .		
jest większa niż 512_{10} .		
jest mniejsza niż 472_8 .		

Komentarz do zadania

Zadanie można rozwiązać na 2 sposoby. Jednym z nich jest zamiana zadanej liczby z systemu binarnego na dziesiętny i sprawdzenie, które odpowiedzi są poprawne:

$$100110010_2 = 1 \cdot 2^8 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^1 = 256 + 32 + 16 + 2 = 306 .$$

Odpowiedź na pytanie pierwsze jest prawdziwa, ponieważ

$$10011001_2 = 1 \cdot 2^7 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^0 = 153.$$

Odpowiedź na pytanie drugie jest prawdziwa, ponieważ

$$1001100100_2 = 1 \cdot 2^9 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^2 = 612.$$

Odpowiedź na pytanie trzecie jest fałszywa, ponieważ liczba 306 jest mniejsza niż 512. W przypadku odpowiedzi na czwarte pytanie można zamienić liczbę zapisaną w systemie ósemkowym na system dziesiętny: $472_8 = 4 \cdot 8^2 + 7 \cdot 8^1 + 2 \cdot 8^0 = 314$, zatem odpowiedź jest prawdziwa.

Przedstawione rozumowanie wymaga jednak poświęcenia na obliczenia sporej ilości czasu. Zadanie można rozwiązać znacznie szybciej, bez zamiany zapisu zadanej liczby z systemu binarnego na dziesiętny.

Po pierwsze, wystarczy wiedzieć, że mnożenie liczby binarnych przez 2^i jest jednoznaczne z dopisywaniem i zer z prawej strony reprezentacji binarnej, zaś dzielenie liczb binarnych przez 2^i jest jednoznaczne z usuwaniem i najmniej znaczących bitów. W naszym przypadku $i = 1$, więc wystarczy:

- usunąć najmniej znaczący bit liczby w celu uzyskania odpowiedzi na pytanie pierwsze: $100110010_2 \rightarrow 10011001_2$,
- dopisać jedno zero z prawej strony liczby w celu uzyskania odpowiedzi na pytanie drugie: $100110010_2 \rightarrow 1001100100_2$

W pytaniu trzecim pojawia się liczba $512_{10} = 2^9 = 100000000_2 > 100110010_2$, zatem odpowiedź jest fałszywa.

W przypadku pytania czwartego wystarczy liczbę binarną rozdzielić na grupy 3-bitowe, idąc od strony prawej ku lewej. Jeśli w ostatniej grupie jest mniej bitów, to brakujące bity uzupełniamy zerami. Następnie każdą z grup bitowych zastępujemy odpowiednią cyfrą ósemkową. W wyniku tego otrzymujemy liczbę ósemkową o identycznej wartości jak wyjściowa liczba binarna: $100110010_2 \rightarrow 100\ 110\ 010_2 \rightarrow 462_8 < 472_8$ (co daje odpowiedź prawdziwą).

Rozwiązanie

PPFP

Zadanie 45.

Wskaż elementy, które są niezbędne do uruchomienia komputera i załadowania systemu operacyjnego.

	P	F
procesor		
twardy dysk		
pamięć operacyjna		
monitor		

Komentarz do zadania

Procesor jest centralną jednostką komputera, która wykonuje wszystkie obliczenia i steruje wykonywaniem instrukcji. Komputer nie może działać bez procesora.

Pamięć operacyjna jest konieczna, aby przechowywać dane, w tym podstawową część (jądro) systemu operacyjnego — komputer bez pamięci nie jest w stanie działać.

Komputer może działać bez trwałego dysku: możliwe jest uruchomienie systemu operacyjnego z płyty CD lub pamięci typu *flash*, a nawet przez sieć lokalną, przy zdalnym użyciu dysku znajdującego się w innym komputerze.

Komputer może bez żadnych przeszkód działać bez monitora. Często na przykład w ten sposób pracują długo działające serwery, które wygodniej obsługiwać zdalnie, logując się na nie z innego komputera.

Zadanie 46.

Zaznacz znakiem X w odpowiedniej kolumnie, które zdanie jest prawdziwe, a które jest fałszywe.

	P	F
System operacyjny przydziela zadaniom czas pracy procesora.		
System operacyjny używa zawsze tego samego systemu plików dla wszystkich urządzeń.		
W skład systemu operacyjnego wchodzi zawsze graficzny interfejs użytkownika.		
System operacyjny przydziela uruchamianym aplikacjom pamięć operacyjną.		

Zadanie 47.

Zaznacz znakiem X w odpowiedniej kolumnie, które zdanie jest prawdziwe (P), a które jest fałszywe (F).

System plików NTFS

	P	F
nie jest obsługiwany przez system Linux.		
przechowuje informację o rozmiarze, dacie utworzenia i modyfikacji pliku oraz o ścieżce dostępu do pliku.		
uniemożliwia zapisanie pojedynczego pliku o rozmiarze powyżej 4 GB.		
umożliwia administratorowi nadawanie pojedynczym użytkownikom lub grupom użytkowników praw dostępu do plików i katalogów.		

Zadanie 48.

Zaznacz znakiem X w odpowiedniej kolumnie, które zdanie jest prawdziwe, a które jest fałszywe.

W pewnej firmie znajdują się m.in. komputery o następujących adresach IP:

- komputer A: 10.20.30.40 / maska 255.255.0.0;
- komputer B: 10.0.0.10 / maska 255.255.255.0;
- komputer C: 1.2.3.4 / maska 255.255.255.0;
- komputer D: 1.2.3.250 / maska 255.255.255.0.

	P	F
Komputer A może być widoczny w sieci Internet pod innym adresem IP.		
Tylko dwa z wymienionych komputerów mogą mieć dostęp do sieci Internet.		
Komputery A i B znajdują się w jednej podsieci.		
Komputery C i D muszą znajdować się w jednym budynku.		

Zadanie 49.

Chmura obliczeniowa jest usługą polegającą na zdalnym udostępnieniu mocy obliczeniowej urządzeń IT, oferowaną przez zewnętrznego dostawcę. Oceń prawdziwość poniższych zdań, umieszczając znak X w odpowiedniej kolumnie tabeli.

		P	F
A	Z aplikacji i danych umieszczonych w chmurze można korzystać z dowolnej lokalizacji i dowolnego sprzętu IT umożliwiającego połączenie internetowe.		
B	Użytkownik nie jest zobowiązany do zakupu licencji na oprogramowanie używane w chmurze i udostępniane przez dostawcę, płaci jedynie za jego użycie (każdorazowo lub w formie abonamentu).		
C	Użytkownik może zdalnie instalować w przydzielonych zasobach chmury dowolne aplikacje i korzystać z nich tak jak na lokalnym komputerze.		
D	Pula zasobów użytkownika (w tym: procesory, pamięć RAM, przestrzeń dyskowa) jest elastycznie skalowana w zależności od jego potrzeb i ograniczona tylko możliwościami dostawcy.		

Zadanie 50.

HTTP Cookie jest niewielką porcją informacji wysyłaną przez witrynę internetową do przeglądarki klienta i zapisywana w jej ustawieniach. Oceń prawdziwość poniższych zdań, umieszczając znak X w odpowiedniej kolumnie tabeli.

		P	F
A	<i>Cookie</i> zawiera polecenia, które konfigurują ustawienia przeglądarki klienta.		
B	<i>Cookie</i> umożliwia serwisowi sprawdzenie, czy klient już go odwiedzał w przeszłości, oraz zapamiętanie upodobań klienta.		
C	<i>Cookie</i> zapisane przez serwis z domeny <i>cwaniak.org</i> może być odczytane przez serwis z domeny <i>spryciarz.org</i> .		
D	Zablokowanie obsługi <i>cookie</i> w przeglądarce może spowodować utrudnienia dla użytkownika dokonującego zakupów w sklepie internetowym.		

Zadanie 51.

Czterech użytkowników założyło konta w pewnym serwisie internetowym. Wyбрали następujące dane logowania:

Użytkownik A	nazwa użytkownika: <i>jankowalski</i> hasło: <i>1234</i>
Użytkownik B	nazwa użytkownika: <i>merkuriusz_312</i> hasło: <i>merkuriusz_312</i>
Użytkownik C	nazwa użytkownika: <i>abc</i> hasło: <i>S4o9s2n5a7</i>
Użytkownik D	nazwa użytkownika: <i>master</i> hasło: <i>password</i>

Przestępca usiłuje przejąć konta użytkowników, stosując w tym celu następujące dwie techniki:

- atak *brute force* (wszystkie możliwe kombinacje znaków),
- metoda psychologiczna i słownikowa (zgadnięcie hasła, sprawdzenie popularnych haseł i słów).

Wskaż, które hasła są podatne na złamanie tymi metodami, stawiając w odpowiednim polu tabeli znak X, jeśli taka podatność występuje:

	atak <i>brute force</i>	metoda słownikowa/psychologiczna
Użytkownik A		
Użytkownik B		
Użytkownik C		
Użytkownik D		

Zadanie 52.

Zaznacz znakiem X w odpowiedniej kolumnie, które zdanie jest prawdziwe (P), a które jest fałszywe (F).

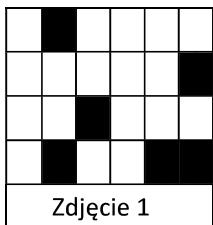
W grafice rastrowej 24-bitowe kodowanie koloru oznacza, że

	P	F
liczba kolorów w palecie barw wynosi ponad 16 milionów.		
informacje o kolorze jednego piksela zajmują 3 bajty.		
liczba kolorów jest niewystarczająca do zapisu zdjęć.		
obraz o rozmiarach 300x300 pikseli zapisany bez kompresji ma wielkość około 2,16 MB.		

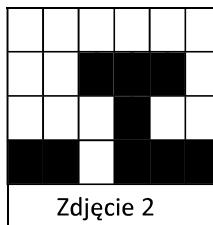
Zadanie 53.

Wiązka zadań Zegar binarny

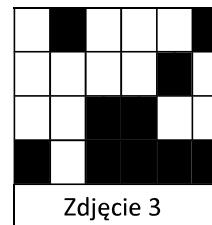
Pan Kowalski postanowił wyjechać na wycieczkę na wyspę Binarną. Mieszkańcy tej wyspy do zapisywania cyfr dziesiętnych używają metody obrazkowej, opartej na systemie binarnym. Podczas pobytu na wyspie Pan Kowalski wybrał się na jednodniową wycieczkę objazdową. W jej trakcie zrobił 4 poniższe zdjęcia zegara używanego na wyspie. Stan zegara podawany jest w formacie GG:MM:SS, gdzie GG to godzina, MM to minuty, a SS to sekundy (zapisane zawsze przy pomocy dwóch cyfr). Każda cyfra zapisana jest w osobnej kolumnie.



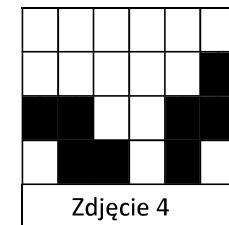
Zdjęcie 1



Zdjęcie 2



Zdjęcie 3



Zdjęcie 4

Pierwsze zdjęcie zostało zrobione w momencie rozpoczęcia wycieczki, ostatnie zdjęcie — w chwili jej zakończenia, zdjęcia 2 i 3 zostały natomiast zrobione w trakcie wycieczki.

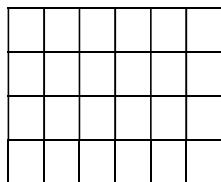
53.1.

Podaj w systemie dziesiętnym stany zegara wyświetlane na zdjęciach 1 oraz 4.

- Stan zegara na zdjęciu nr 1:
- Stan zegara na zdjęciu nr 4:

53.2.

Pan Kowalski położył się spać o godzinie 23:45:29. Zamaluj odpowiednie pola poniższej planszy tak, aby reprezentowała ona tę godzinę na zegarze używanym przez mieszkańców wyspy.



Zadanie 54.

Zaznacz znakiem X w odpowiedniej kolumnie, które zdanie jest prawdziwe (P), a które jest fałszywe (F).

Stos jest strukturą danych, która umożliwia

	P	F
bezpośredni dostęp do ostatnio zapisanego elementu.		
bezpośredni dostęp do każdego elementu stosu.		
bezpośredni dostęp do najmniejszego i największego elementu stosu.		
dodanie nowego elementu oraz usunięcie najpóźniej dodanego elementu.		

Zadanie 55.

Wiązka zadań *Zadania Zamknięte Funkcja*

Dana jest następujący algorytm $F(n)$ dla $n \in N, n > 0$:

$F(n)$

jeżeli $n = 1$, zwróć 1 i zakończ
w przeciwnym razie zwróć $F(n \text{ div } 2) + 1$

55.1.

Złożoność tego algorytmu jest

	P	F
wykładnicza.		
logarytmiczna.		
liniowa.		
kwadratowa.		

55.2.

Dla tego algorytmu zachodzi

	P	F
$F(8) = 3$.		
$F(12) = 4$.		
$F(1) = 0$ lub $F(9) = 4$.		
$F(1) = 1$ oraz $F(9) = 3$.		

Zadanie 56.

Piractwo komputerowe jest przestępstwem polegającym na:

	P	F
wykorzystywaniu oprogramowania w celu osiągnięcia korzyści majątkowej, bez licencji na jego użytkowanie.		
instalowaniu wersji demo oprogramowania, którego licencję planujemy kupić.		
rozpowszechnianiu w Internecie programów komputerowych, których licencję zakupiliśmy.		
ofierowaniu za darmo w Internecie oprogramowania rozpowszechnianego na licencji freeware.		

Zadanie 57.

Zgodnie z prawem w Internecie można opublikować zdjęcie osoby:

	P	F
po uzyskaniu od niej zezwolenia.		
gdy jest to osoba powszechnie znana i zdjęcie zostało wykonane podczas pełnienia przez nią funkcji publicznych, w szczególności politycznych, społecznych, zawodowych.		
gdy osoba ta jest naszym bliskim znajomym.		
gdy stanowi ona jedynie szczegół całości takiej jak: zgromadzenie, krajobraz, publiczna impreza.		