Rekurencja w zadaniach maturalnych

Zadanie 1.

Opisana poniżej funkcja rekurencyjna wyznacza, dla liczby naturalnej $n > 0$, długość napisu uzyskanego przez sklejenie binarnych reprezentacji liczb naturalnych od 1 do $n-1$.

Funkcja $sklej(n)$

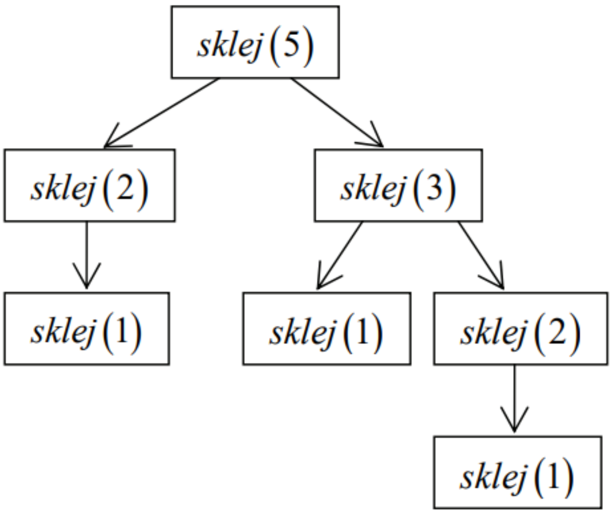
krok 1. jeśli $n = 1$, to podaj 0 jako wynik i zakończ działanie

krok 2. jeśli n parzysta, to wynikiem jest $n - 1 + 2 \cdot sklej(n/2)$

krok 3. jeśli n nieparzysta, to wynikiem jest $n - 1 + sklej((n-1)/2) + sklej((n+1)/2)$

Wykonaj polecenia a)–c):

a) Wykonanie funkcji *sklej* można przedstawić w postaci drzewa wywołań rekurencyjnych ilustrującego wszystkie wywołania funkcji po jej uruchomieniu dla zadanego argumentu. Poniższy rysunek przedstawia takie drzewo dla wywołania $sklej(5)$.



Narysuj analogiczne drzewo dla wywołania $sklej(7)$.

b) Uzupełnij poniższą tabelę, podając wartości funkcji *sklej* dla wskazanych argumentów.

n	$sklej(n)$
1	0
2	1
3	
4	
5	
6	

c) Chcemy wypełnić tablicę $s[1..n]$ w taki sposób, że $s[i] = sklej(i)$ dla każdego $1 \le i \le n$. Podaj algorytm wypełniający tablicę s odpowiednimi wartościami **bez wywoływania** funkcji *sklej*, tzn. **bez** użycia **rekurencji**. Zauważ, że jeśli poprawnie wyliczone są już wartości $s[1], ..., s[i-1]$, to można z nich skorzystać przy wyznaczaniu $s[i]$.

Zapisz swój algorytm w postaci listy kroków, schematu blokowego lub w wybranym języku programowania, który wybrałeś/aś na egzamin.

Specyfikacja:
Dane: liczba naturalna $n > 0$
Wynik: tablica $s[1..n]$ o wartościach $s[i] = sklej(i)$, dla $1 \le i \le n$

Algorytm:

Zadanie 2.

Dana jest liczba naturalna $n > 0$ i tablica różnych liczb całkowitych $a[1..n]$. Rozważamy następującą **rekurencyjną** funkcję F z argumentem i będącym liczbą naturalną, $1 \le i \le n$.

Funkcja $F(i)$
    jeżeli $i = n$ to
        wynikiem jest n
    w przeciwnym razie
        $j := F(i+1)$
        jeżeli $a[i] < a[j]$ wtedy
            wynikiem jest i
        w przeciwnym razie
            wynikiem jest j

a) Dla danej 10-elementowej tablicy $a = [5, 1, 8, 9, 7, 2, 3, 11, 20, 15]$ podaj w poniższej tabeli wynik wywołania funkcji F dla danego argumentu i.

i	$F(i)$
9	
7	
5	

b) Niech w będzie wynikiem wywołania funkcji F dla argumentu i, $1 \le i \le n$. Wtedy $a[w]$ w odniesieniu do pozostałych liczb w tablicy a jest zawsze

- najmniejszą liczbą w tej tablicy.
- najmniejszą liczbą w tej tablicy spośród elementów o indeksach od i do n.
- najmniejszą liczbą w tej tablicy spośród elementów o indeksach od 1 do i.

Podkreśl właściwą odpowiedź.

c) Ile porównań między elementami tablicy zostanie wykonanych przy wywołaniu $F(512)$ dla $n = 2012$?

d) Zapisz funkcję F **iteracyjnie**.

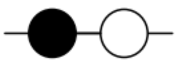
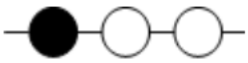
Zadanie 3.

Rozważamy następującą **rekurencyjną** procedurę ***Korale***, której parametrem jest dodatnia liczba całkowita n.

Korale(n)

1. Jeżeli $n = 1$, to
   1.1. nawlecz czarny koralik na prawy koniec sznurka,
   1.2. zakończ działanie procedury.
2. Jeżeli n jest parzyste, to
   2.1. wykonaj ***Korale(n/2)***,
   2.2. nawlecz biały koralik na prawy koniec sznurka,
   2.3. zakończ działanie procedury.
3. Jeżeli n jest nieparzyste, to
   3.1. wykonaj ***Korale((n-1)/2)***,
   3.2. nawlecz czarny koralik na prawy koniec sznurka,
   3.3. zakończ działanie procedury.

a) Uzupełnij tabelę i w ten sposób przedstaw wynik działania powyższego algorytmu dla podanych argumentów n:

n	wynik działania **Korale(*n*)**
1	
2	
3	
4	
7	
8	
15	
16	

b) Ile koralików zostanie nawleczonych na sznurek w wyniku wywołania procedury ***Korale*** dla danej liczby n? Odpowiedź uzasadnij.

c) Zaprojektuj i zapisz nierekurencyjną procedurę ***KoraleBis(n)***, po wykonaniu której uzyskamy taki sam efekt, jak po wykonaniu ***Korale(n)***. W procedurze ***KoraleBis*** można nawlekać koraliki tylko na jeden, wybrany koniec sznurka.

Algorytm:

Zadanie 4.

Dana jest funkcja f określona wzorem rekurencyjnym

$$\begin{cases} f(1) = 4 \\ f(n+1) = \dfrac{1}{1-f(n)} & \text{dla } n \geq 1 \end{cases}$$

Wtedy:

1.	$f(8) = \frac{1}{3}$	**P**	**F**
2.	$f(9) = \frac{3}{4}$	**P**	**F**
3.	$f(10) = 4$	**P**	**F**
4.	$f(100) = -\frac{1}{3}$	**P**	**F**

Zadanie 5.

Rozważ następujący algorytm zapisany w postaci rekurencyjnej funkcji F:

Specyfikacja:

Dane:

n – liczba całkowita dodatnia

Algorytm:

***F*(*n*)**

Jeżeli n=1 lub n=2

$s \leftarrow n$

w przeciwnym razie

$s \leftarrow n$****F***(n–2)

$s \leftarrow s$*(n+1)

wynikiem jest s

Zadanie 5.1.

Uzupełnij poniższą tabelę – podaj wartości funkcji dla n=1, 2, 3, 4, 5, 6.

n	$F(n)$
1	
2	
3	
4	
5	
6	

Zadanie 5.2.

Wypisz ciąg wywołań funkcji $F(n)$ dla n=11.

Przykład:

Dla n=3 ciąg wywołań ma postać $F(3)$, $F(1)$.

....

Zadanie 5.3.

Podaj wynik działania algorytmu – zaznacz prawidłową odpowiedź.

Algorytm obliczy wartość:

a) $\frac{(n+1)(n+2)}{2}$

b) $(n+1)!$

c) $n^{\frac{n}{2}}$

Zadanie 6.

Rozważmy ciąg liczb $p_0, p_1, p_2, \ldots$ zdefiniowany w następujący sposób:

$$\begin{cases} p_0 = 0 \\ p_1 = 1 \\ p_2 = 1 \\ p_3 = 2 \\ p_4 = 4 \\ p_n = p_{n-1} + p_{n-2} + p_{n-3} + p_{n-4} + p_{n-5} \; dla \; n \geq 5 \end{cases}$$

Zadanie 6.1.

Uzupełnij poniższą tabelę.

n	p_n
5	8
7	
9	

Zadanie 6.2.

Poniżej prezentujemy algorytm, który powinien wyznaczać n-ty element podanego ciągu. Uzupełnij luki w algorytmie tak, aby jego działanie było zgodne z podaną specyfikacją.

Specyfikacja:

Dane: n – nieujemna liczba całkowita
Wynik: w – liczba całkowita równa p_n

Algorytm:

```
tab[0] ← 0
tab[1] ← 1
tab[2] ← 1
tab[3] ← 2
tab[4] ← 4
i ← 5
dopóki i ≤ .... wykonuj
      temp ← tab[0]+tab[1]+tab[2]+tab[3]+tab[4]
      tab[..... mod 5] ← temp
      i ← i+1
w ← ...............
```

Uwaga: $a \bmod b$ oznacza resztę z dzielenia liczby a przez liczbę b.

Zadanie 6.3.

Rozważmy poniższy ciąg r_n:

$$\begin{cases} r_0 = 0 \\ r_1 = 1 \\ r_2 = 1 \\ r_3 = 0 \\ r_4 = 0 \\ r_n = (r_{n-1} + r_{n-2} + r_{n-3} + r_{n-4} + r_{n-5})\ mod\ 2\ dla\ n \geq 5 \end{cases}$$

Zauważmy, że liczba p_n jest parzysta wtedy i tylko wtedy, gdy r_n=0. Można też sprawdzić, że wartości r_n powtarzają się cyklicznie – każda wartość jest taka sama jak wartość wcześniejsza o sześć wyrazów – a zatem wartość r_n zależy wyłącznie od liczby n mod 6. Na podstawie tego faktu podaj algorytm **o jak najmniejszej złożoności obliczeniowej**, który działa zgodnie z poniższą specyfikacją.

Specyfikacja:

Dane: n – nieujemna liczba całkowita
Wynik: w – 0 (zero), gdy liczba p_n jest parzysta, natomiast 1 (jeden), gdy liczba p_n jest nieparzysta

Zadanie 7.

Funkcja `licz(x)` przyjmuje jako argument dodatnią liczbę całkowitą x, natomiast jako wynik daje pewną liczbę całkowitą.

```
licz(x)
     jeżeli x = 1
        podaj wynik 1
     w przeciwnym przypadku
        w ← licz(x div 2)
        jeżeli x mod 2 = 1
           podaj wynik w+1
        w przeciwnym przypadku
           podaj wynik w-1
```

Uwaga: *div* – dzielenie całkowite, *mod* – reszta z dzielenia całkowitego.

Zadanie 7.1.

Uzupełnij tabelę – podaj wartość `licz(x)` dla podanych argumentów x.

x	`licz(x)`
11	2
13	
21	
32	

Zadanie 7.2.

Dana jest dodatnia liczba całkowita k. Jaka jest najmniejsza dodatnia liczba całkowita x, dla której obliczanie wartości `licz(x)` wymaga dokładnie k wywołań funkcji `licz`, licząc także pierwsze wywołanie `licz(x)`? Podkreśl prawidłową odpowiedź.

Przykład: obliczenie `licz(13)` wymaga dokładnie 4 wywołań funkcji `licz`.

A) $x = k^2$

B) $x = 2^{k-1}$

C) $x = k+1$

D) $x = 2^k$

Zadanie 7.3.

Podaj najmniejszą liczbę całkowitą x większą od 100, dla której wynikiem wywołania `licz(x)` będzie 0.

Odpowiedź: ________________

Zadanie 8.

Liczby Fibonacciego są definiowane w następujący sposób:

$$F_1 = 1, \quad F_2 = 1,$$

$$F_n = F_{n-1} + F_{n-2} \text{ dla } n = 3, 4, \ldots$$

Rekurencyjny algorytm, który służy do obliczania wartości F_n dla dowolnego $n \geq 1$, można zapisać następująco:

```
funkcja F(n)
  jeśli n=1 lub n=2
      wynikiem jest 1
  w przeciwnym razie
      wynikiem jest F(n-1) + F(n-2)
```

Zadanie 8.1.

Zapisz w wybranej przez siebie notacji (w języku programowania lub w pseudokodzie) algorytm iteracyjny, który służy do obliczania wartości liczby F_n dla dowolnego $n \geq 1$. Algorytm nie może używać tablic.

Zadanie 8.2.

Aby obliczyć F_{45}, wywołano najpierw funkcję iteracyjną, a potem – rekurencyjną. Okazało się, że czas trwania obliczeń realizowanych przez funkcję rekurencyjną był długi, podczas gdy funkcja iteracyjna prawie natychmiast podała wynik. Uzasadnij długi czas działania funkcji **rekurencyjnej**.

Zadanie 8.3.

Aby przyśpieszyć rekurencyjne obliczanie wartości n-tego wyrazu ciągu Fibonacciego, można skorzystać z następujących wzorów, prawdziwych dla dowolnego całkowitego $k \geq 2$:

$$F_{2k} = (F_{k+1})^2 - (F_{k-1})^2$$

$$F_{2k-1} = (F_k)^2 + (F_{k-1})^2$$

Zapisz w wybranej przez siebie notacji (w postaci listy kroków, w języku programowania lub w pseudokodzie) algorytm **rekurencyjny**, który służy do obliczania wartości liczby F_n dla dowolnego $n \geq 1$ i korzysta z tych wzorów.

Zadanie 9.

Dana jest funkcja rekurencyjna *Rek*, której argumentem jest nieujemna liczba całkowita *n*.

```
funkcja Rek(n)
   jeśli (n>0) to wykonaj kolejno dwie instrukcje:
          1. wywołaj Rek dla argumentu n-1
          2. wypisz n
```

Jeśli wywołamy ją dla *n* równego 5, to:

1	Zero będzie wypisane.	**P**	**F**
2	Największą wypisaną liczbą będzie 5.	**P**	**F**
3	Zostanie wypisanych 5 liczb.	**P**	**F**
4	Liczby zostaną wypisane w kolejności malejącej.	**P**	**F**

Zadanie 10.

Przeanalizuj podaną funkcję `pisz`.

Specyfikacja:

Dane:

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

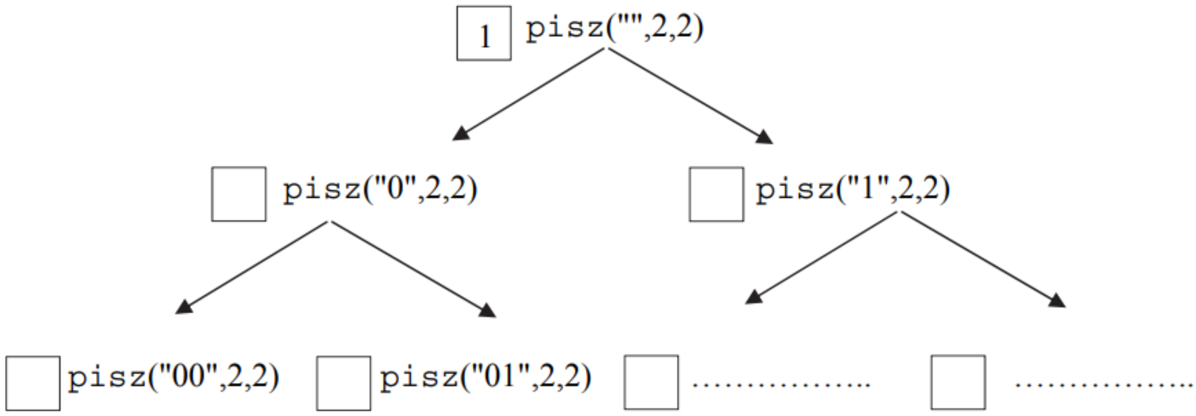
```
funkcja pisz(s,n,k)
     jeżeli dł(s) = n
         wypisz s
     w przeciwnym razie
          dla i=0,1 … k-1 wykonuj
               pisz(s + napis(i), n, k)
```

Uwaga:
dł(*x*) – daje w wyniku długość napisu *x*
s1 + *s2* – daje w wyniku złączenie napisów *s1* i *s2*
napis(*p*) – daje w wyniku napis będący zapisem dziesiętnym liczby całkowitej *p*

Zadanie 10.1.

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

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



Zadanie 10.2.

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

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

Zadanie 10.3.

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

Zadanie 11.

Dana jest dodatnia liczba całkowita n oraz uporządkowana rosnąco tablica różnych liczb całkowitych $T[1..n]$. Przeanalizuj następującą funkcję *rekurencyjną*, której parametrami są liczby całkowite x, p, k, przy czym $1 \le p \le k \le n$.

Rek(x, p, k)
jeżeli $p < k$
    $s \leftarrow (p + k)$ div 2
    jeżeli $T[s] \ge x$
        wynikiem jest ***Rek***(x, p, s)
    w przeciwnym razie
        wynikiem jest ***Rek***$(x, s + 1, k)$
w przeciwnym razie
    jeżeli $T[\mathrm{p}] = x$
        wynikiem jest p
    w przeciwnym razie
        wynikiem jest -1

Uwaga: **div** jest operatorem oznaczającym część całkowitą z dzielenia.

Zadanie 11.1.

Podaj największą i najmniejszą możliwą liczbę wywołań funkcji *Rek* w wyniku wywołania ***Rek***(2019, 6, 14) dla n = 17 i pewnej, uporządkowanej rosnąco tablicy T[1..17] różnych liczb całkowitych.

Uwaga: Pierwsze wywołanie funkcji ***Rek***(2019, 6, 14) włączamy do ogólnej liczby wywołań.

Zadanie 11.2.

Podaj, jakie będą wartości parametrów przekazywanych do funkcji *Rek* w kolejnych jej wywołaniach dla $n = 11$, tablicy $T = [1, 5, 8, 10, 12, 14, 19, 20, 23, 30, 38]$ oraz pierwszego wywołania ***Rek*****(37, 1, 11)**.

Zadanie 11.3.

Złożoność czasowa algorytmu opisanego funkcją ***Rek*** dla parametrów $x = 1$, $p = 1$, $k = n$ jest

A. sześcienna.

B. kwadratowa.

C. liniowa.

D. logarytmiczna.

Wybierz właściwą odpowiedź.

Zadanie 12.

Argumentami procedury sym (a, b) są dwie nieujemne liczby całkowite a i b. Wywołanie tej procedury spowoduje wypisanie pewnego ciągu liczb całkowitych.

```
sym(a,b)
    jeżeli a ≠ 0
        sym(a – 1,  b + 1)
        wypisz a * b
        sym(a – 1,  b + 1)
```

Zadanie 12.1.

Uzupełnij tabelę – podaj wynik działania procedury sym (a, b) dla wskazanych argumentów a i b.

a	*b*	**sym (*a*, *b*)**
3	1	3 4 3 3 3 4 3
4	2	5 8 5 9 5 8 5 8 5 8 5 9 5 8 5
3	3	
4	1	

Zadanie 12.2.

Uzupełnij tabelę – podaj długość ciągu liczbowego otrzymanego w wyniku wywołania procedury sym (a, b) dla wskazanych argumentów a i b.

a	*b*	**sym (*a*, *b*)**
3	2	7
4	4	15
5	1	
6	6	
10	2020	

Zadanie 13.

Dana jest następująca funkcja:

funkcja *f*(*n*):
  jeżeli *n* > 0
    wypisz *n*
    f(*n* **–** 2)
    wypisz *n*

1.	W wyniku wywołania *f*(5) otrzymamy ciąg 5 5 5 5 5 5.	**P**	**F**
2.	W wyniku wywołania *f*(6) otrzymamy ciąg 6 4 2 2 4 6.	**P**	**F**
3.	W wyniku wywołania *f*(7) otrzymamy ciąg 7 5 3 1 1 3 5 7.	**P**	**F**
4.	W wyniku wywołania *f*(8) otrzymamy ciąg 8 6 4 2 0 0 2 4 6 8.	**P**	**F**