









**Zadanie 3.3.** Egzamin maj 2014 r. Arkusz I, poziom rozszerzony,  
zadanie 1. KORALE

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 <i>Korale</i> ( $n$ )
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.

W wyniku wywołania procedury *Korale()* dla danej liczby  $n$  zostanie nawleczonych na sznurek  $\log_2(n + 1)$  koralików.

**Uzasadnienie:**

Sznurki koralików można potraktować jako rozwinięcie bitowe liczby  $n$  przy założeniu, że koralik czarny odpowiada bitowi 1, a koralik biały bitowi 0. Największa liczba  $n$ , jaką możemy przedstawić za pomocą  $x$  koralików, wynosi  $2^x - 1$ , z czego wynika  $2^x - 1 = n$ , a po przekształceniu  $2^x = n + 1$ , czyli  $x = \log_2(n + 1)$ .

- 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.

**Listing (*zad\_c.py*):**

```
def zad_c(n):
    s = ''
    while n > 0:
        if n % 2 == 0: s = 'B' + s
        else: s = 'C' + s
        n //= 2
    return s

print(zad_c(16))
```