Zadanie 1. Fibonacci *(7 pkt)*

Poniższa funkcja rekurencyjna *Fib* oblicza *k*-ty wyraz ciągu Fibonacciego.

Dane: k – liczba naturalna większa od zera

Funkcja $Fib(k)$

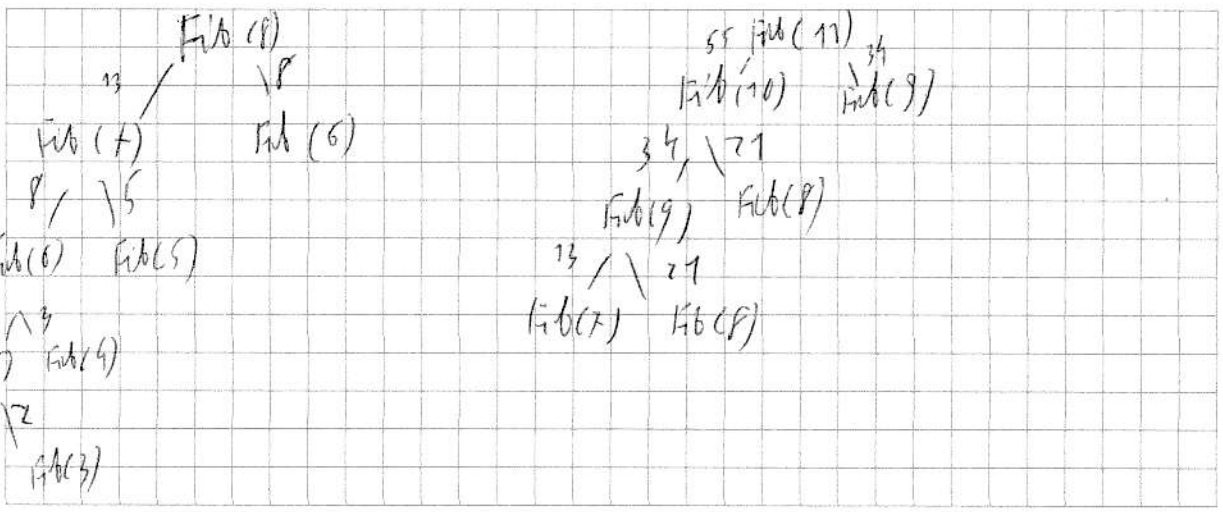
1. Jeżeli $k=1$ lub $k=2$, to wynikiem jest 1.
2. Jeżeli $k>2$, to wynikiem jest $Fib(k-1)+Fib(k-2)$.

Przykład:
Zgodnie z powyższą definicją funkcji *Fib* mamy:

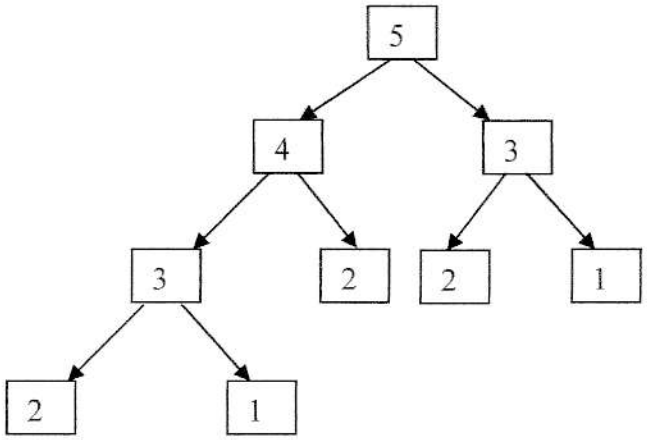
$$\begin{aligned} Fib(4) &= Fib(3) + Fib(2) = \\ &= [Fib(2)+Fib(1)]+Fib(2) = \\ &= [\ 1\ +\ 1\ ]+\ 1\ = 3 \end{aligned}$$

a) Uzupełnij tabelę, wpisując dla podanych argumentów *k* wartości obliczane przez funkcję *Fib*.

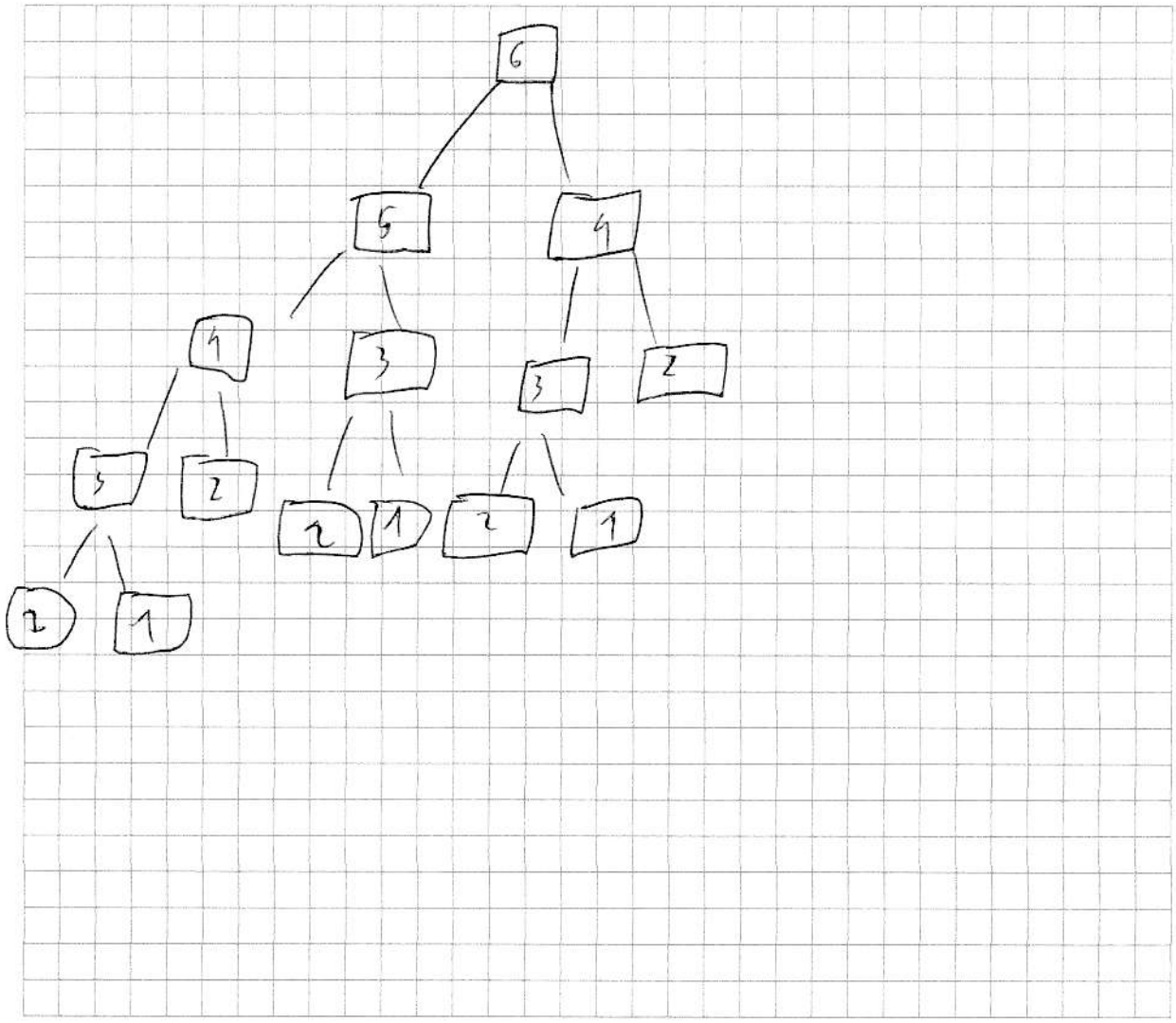
k	$Fib(k)$
1	1
2	1
3	2
...	...
8	21
...	...
11	89



b) Wywołanie funkcji $Fib(k)$ dla $k > 2$ powoduje dwa kolejne wywołania tej funkcji z mniejszymi argumentami, które z kolei mogą wymagać kolejnych wywołań Fib, itd. Proces ten można zilustrować za pomocą tzw. drzewa wywołań rekurencyjnych. Poniżej prezentujemy drzewo wywołań rekurencyjnych dla $k = 5$. W węzłach drzewa znajdują się argumenty wywołań.



Narysuj drzewo wywołań rekurencyjnych dla $Fib(6)$.



c) k-ty wyraz ciągu Fibonacciego można wyznaczyć iteracyjnie w następujący sposób:

Dane: k – liczba naturalna większa od zera

Algorytm:

1. $Fi \leftarrow 1$, $Fi\_1 \leftarrow 1$, $i \leftarrow 2$
2. dopóki $i < k$

   $pom \leftarrow Fi$

   $Fi \leftarrow Fi + Fi\_1$

   $Fi\_1 \leftarrow pom$

   $i \leftarrow i+1$
3. wypisz Fi

Zdefiniujmy następujący ciąg:
- Pierwszy i drugi wyraz ciągu są równe 1.
- Jeśli $k > 2$ i k jest parzyste, to k-ty wyraz jest sumą trzech wyrazów go poprzedzających.
- Jeśli $k > 2$ i k jest nieparzyste, to k-ty wyraz jest równy wyrazowi o numerze $(k-1)$.

Kilka pierwszych wyrazów tego ciągu podano w poniższej tabeli.

k	1	2	3	4	5	6	7	8
k-ty wyraz	1	1	1	3	3	7	7	17

Zapisz algorytm (w postaci listy kroków, schematu blokowego lub w wybranym języku programowania), który dla danej wartości k wyznacza k-ty wyraz opisanego powyżej ciągu. Zapisz rozwiązanie w postaci **iteracyjnej**.

Specyfikacja:

Dane: k – liczba naturalna większa od zera

Wynik: k-ty wyraz ciągu zdefiniowanego powyżej

Algorytm:

```
K[1] = 1
K[2] = 1
for(i=3; i<=k; i++)
  { if (i%2 == 0) { K[i] = K[i-1] + K[i-2]; }
    else { K[i] = K[i-1]; }
Cout << K[i] << endl;
```