

# POPRAWNE ODPOWIEDZI DO TESTU

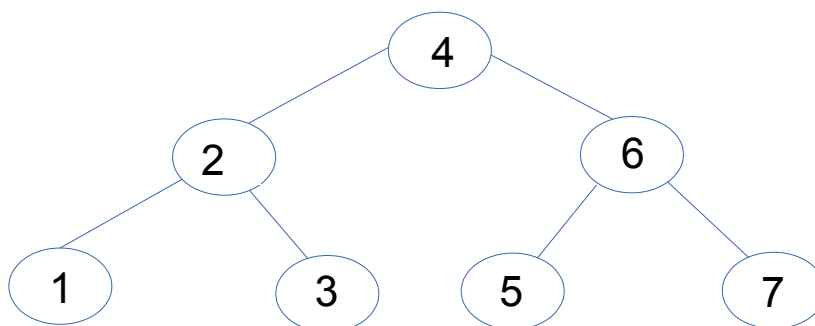
[1] D - PUSH( ) Jeżeli stos jest zapełniony, to nie damy już rady wstawić kolejnego elementu do pojemnika.

[2] B - głowę, czyli pierwszy element ustawiony w kolejce. Ewentualnie można to też nazwać wierzchołkiem, chociaż to nazwa używana raczej w kontekście stosu, niż kolejki.

[3] B - 4 elementy. Zwróćmy uwagę, iż w treści zadania wyraźnie zaznaczono: *Dany jest stos, który może przechować maksymalnie cztery liczby całkowite*. Zatem na początku wstawiono na stos liczbę 7, potem zdjęto 7, zdjęto 13, wstawiono 23, wstawiono 91, wstawienie 18 nie powiodło się, bo stos był już zapełniony, a na koniec wywołano funkcję empty (czyli sprawdzającą, czy stos jest pusty - zwróciła ona wartość false). Po tych wszystkich operacjach na stosie mamy 4 liczby.

[4] C - zdejmujemy z listy pierwszą jedynekę, sortujemy całość rosnąco, po czym odwracamy kolejność elementów na liście, uzyskując malejące ułożenie pokazane w zadaniu.

[5] W treści zadania powiedziano: *Wstaw kolejno liczby* - zatem zachowując regułę drzewa otrzymamy następujące drzewo:



[6] C - FOLI. Co prawda kolejność wyrażeń jest odwrócona, ale jeżeli pierwszy na wyjściu (**F**irst **O**ut) będzie element, który ostatni wszedł do pojemnika (**L**ast **I**n), to jest to prawidłowe opisanie zasady działania stosu, (w przeciwieństwie do pozostałych akronimów, prawda).

[7] A - stosu. Pierwsza weszła dwunastka, jako druga w pojemniku znalazła się wartość 7. Jako pierwsza strukturę opuściła siódemka, co oznacza że mamy do czynienia z architekturą LIFO (ostatni, który wszedł, był pierwszym na wyjściu), a więc jest to stos, a nie kolejka.

[8] Węzeł z wartością (-2.44) nie spełnia reguły drzewa, ponieważ znajduje się po lewej stronie swojego rodzica (-2.77), a przechowuje wartość większą niż ma w sobie rodzic ( $-2.44 > -2.77$ , bo -2.44 jest „mniej na minusie” na osi liczbowej).