

Zadania na tej liście dotyczą drzew binarnych tworzonych z użyciem następujących definicji typów lub konstruktorów:

Język C:

```
typedef struct node *pnode;
typedef struct node{
    int val;
    pnode left;
    pnode right;} snode;
```

Język Python:

```
class TreeItem:
    def __init__(self,value):
        self.val = value
        self.left = None
        self.right = None
```

W rozwiązaniach zadań w języku C można korzystać z funkcji

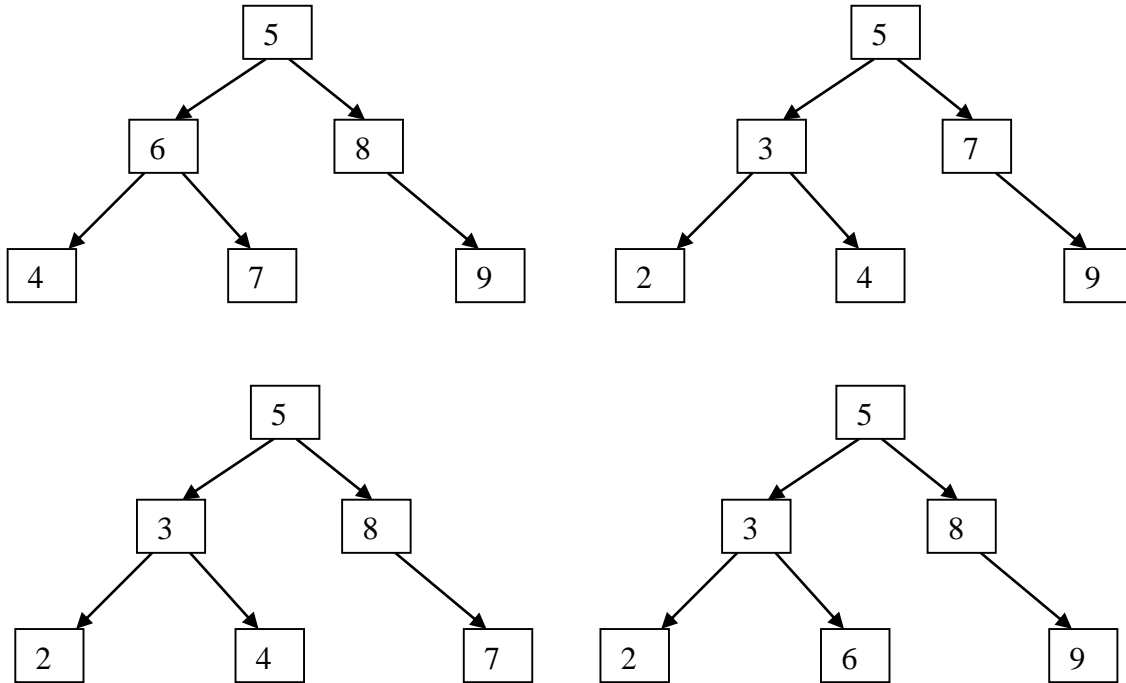
```
pnode utworz(int wart)
```

tworzącej drzewo z jednym węzłem z wartością `val` równą `wart` (p. notatka do wykładu).

- [1] Do drzewa BST (na początku pustego) wstawiane są elementy 1, 2, 3, 4, 5, 6, 7. Podaj kolejność wstawiania elementów, przy której drzewo będzie miało największą / najmniejszą możliwą wysokość. Odpowiedź uzasadnij i uogólnij na przypadek ciągu liczb $1, 2, \dots, 2^k - 1$ dla dowolnego naturalnego $k > 1$.
- [1] Napisz funkcję, która dla parametru t wskazującego na korzeń drzewa binarnego zwraca jako wartość liczbę elementów w drzewie o korzeniu t .
- [1] Napisz funkcję, która dla parametru t zwraca jako wartość *wysokość* drzewa o korzeniu t .
- [0.5] Napisz funkcję, która dla parametru t opisującego drzewo BST wypisuje (w porządku niemalejącym) wszystkie elementy dodatnie znajdujące się w drzewie o korzeniu t .
- [2] Napisz funkcję, która dla danego drzewa binarnego sprawdza czy jest ono drzewem BST.
- [1] Napisz funkcję, która łączy dwa drzewa BST w jedno drzewo przy założeniu, że największa wartość klucza (`val`) w pierwszym drzewie jest mniejsza od najmniejszej wartości klucza w drugim drzewie. Czas działania Twojej funkcji powinien być $O(h)$, gdzie h to wysokość pierwszego drzewa.
- [1] Napisz funkcję wstawiającą element o podanym kluczu do drzewa BST bez użycia rekurencji.
- [1] Rotacją nazywamy operację przebudowy drzewa BST tak, aby wskazany węzeł u i jego wskazane dziecko v „zmieniły miejsca” w taki sposób, że u stanie się dzieckiem v . Podaj sposób na wykonanie rotacji w sytuacji, gdy v jest lewym dzieckiem u .

Zadania dodatkowe, nieobowiązkowe (nie wliczają się do puli punktów do zdobycia na ćwiczeniach, punktacja została podana tylko jako informacja o trudności zadań wg wykładowcy)

9. [0] Które z poniższych drzew są drzewami BST:



10. [0.5] Opisz efekt działania poniższych funkcji dla dowolnego drzewa binarnego t i dla drzewa BST:

```

wypisz(pnode t)
{ if (t!=NULL){
    printf("%d\n", t->val);
    wypisz(t->left);
    wypisz(t->right);
}
}
  
```

```

def wypisz(t):
    if (t!=None)
        print t.val
        wypisz(t.left)
        wypisz(t.right)
  
```

```

wypisz(pnode t)
{ if (t!=NULL){
    wypisz(t->left);
    printf("%d\n", t->val);
    wypisz(t->right);
}
}
  
```

```

def wypisz(t):
    if (t!=None)
        wypisz(t.left)
        print t.val
        wypisz(t.right)
  
```

11. [2] Napisz funkcję, która dla parametru t opisującego drzewo BST wypisuje (w porządku rosnącym) wszystkie elementy dodatnie znajdujące się w drzewie o korzeniu t . Czas działania Twojej funkcji powinien być $O(h+m)$, gdzie h to wysokość drzewa t , a m to liczba elementów dodatnich w drzewie.
12. [0] Napisz funkcję wyszukującą element o podanym kluczu w drzewie BST bez użycia rekurencji.

13. [1] Napisz funkcję usuwającą element o podanym kluczu z drzewa BST bez użycia rekurencji.
14. [2] Napisz funkcję, która dla parametru t opisującego drzewo BST wypisuje w porządku niemalejącym wszystkie elementy dodatnie znajdujące się w drzewie o korzeniu t , bez użycia rekurencji.
15. [1] Dla funkcji wyszukiującej element w drzewie BST utwórz wersję, w której operacja ta będzie przyspieszona przez użycie wartownika.

Wskazówka: wszystkie wskaźniki NULL/None zastąp wskaźnikami do jednego węzła, w którym umieszczasz szukany klucz przed przystąpieniem do wyszukiwania.