

❖ Készítsünk rekurzív algoritmust, mely egy láncoltan ábrázolt bináris fa leveleiben található kulcsok minimumát adja vissza. Üres fára jelezzon hibát.

MinKulcs(t:BinTree):T

t = 0	
Error	return(minKeresés(t))

minKeresés(t:BinTree):T

t → left = 0 és t → right = 0			
return t → key	t → left ≠ 0 és t → right ≠ 0	t → left ≠ 0 és t → right = 0	t → left = 0 és t → right ≠ 0
	min1:=minKeresés(t→left) min2:=minKeresés(t→right) min1:=min(min1,min2)	min1:=minKeresés(t→ left)	min1:=minKeresés(t→ right)
	return min(min1, t→key)		

\* \* \* \* \*

❖ Adott egy láncoltan ábrázolt bináris fa. Készítsen rekurzív algoritmust (nem szintfolytonos bejárással), mely megadja azt a legkisebb szintszámot, ahol a fának levele van. Üres fára -1-et kell visszaadnia.

kisebbSZINT(t:binTree):N

t = 0	
return -1	k := 0
	return(minSZkeresés(t) , k)

minSZkeresés(t:binTree , k) : N

t → left = 0 és t → right = 0			
return k	t → left ≠ 0 és t → right ≠ 0	t → left = 0 és t → right ≠ 0	t → left ≠ 0 és t → right = 0
	k1:= minSZkeresés(t→left,k+1) k2:= minSZkeresés(t→right,k+1) k1:= min(k1,k2)	k1:= minSZkeresés(t→right,k+1)	k1:= minSZkeresés(t→left,k+1)
	return k1		

\* \* \* \* \*

◆ Adott két láncoltan ábrázolt bináris fa t1 és t2. Készítsünk rekurzív logikai függvényt, mely eldönti, hogy a két fa alkja megegyezik-e. A tárolt kulcsoknak nem kell egyenlőnek lenniük, csak az alakjuknak!

**megegyezikE(t1:binTree,t2:binTree): L**

t1 =0 és t2 = 0	
return true	return(hasonlítás(t1,t2))

**hasonlítás(t1:binTree,t2:binTree): L**

t2→ left = 0 és t2→ right = 0 és t1→ left = 0 és t1→ right = 0			
return true	t1→ left ≠ 0 és t1→ right ≠ 0 és t2→ left ≠ 0 és t2→ right ≠ 0		
	t1→ left = 0 és t1→ right ≠ 0 és t2→ left = 0 és t2→ right ≠ 0		
	t1→ left ≠ 0 és t1→ right=0 és t2→ left ≠ 0 és t2→ right=0		
	return(hasonlítás(t1→right,t2→right))	return(hasonlítás(t1→left,t2→left))	return false

l1:= hasonlítás(t1→left,t2→left)  
l2:= hasonlítás(t1→right,t2→right)  
return(l1 és l2)