In il codice colorato di **rosso**, è il codice della funzione ricorsiva copiato così com'è

```
1)  IterativeFun(x, i, j)
2)      Stack  S_x, S_j, S_hL, S_hR, S_zL  // Dichiarazione degli Stack individuati
3)      call ← true
4)      while (call ∨ isNotEmpty(S_x)) do  // Scorre finché è indiscesa o c'è ancora qualcosa nello stack
5)          if call then  // La funzione è in discesa (call = true)
6)              a ← F_ini(i, j)
7)              if x = ⊥ then
8)                  m ← F_⊥(a)
9)                  z ← F_fin(m)
10)                 ret ← z
11)                 (call, last) ← (false, x)  // metto call a false perché ho incontrato un return
                                                // salvo in last l'ultimo nodo letto, ovvero x
12)             else
13)                 (k_L, h_L) ← F_pre(x, a)
14)                 S_x ← Push(S_x, x)        // Essendo arrivati ad una chiamata ricorsiva (la prima),
15)                 S_j ← Push(S_j, j)        // faccio il push negli stack delle variabili
16)                 S_hL ← Push(S_hL, h_L)    // che ne hanno bisogno e che sono state utilizzate fin'ora
17)                 (x, j) ← (x.sx, k_L)      //  Sostituisco le variabili dei parametri, con gli argomenti
                                                  della chiamata
18)         else // La funzione è in risalita (call = false)
19)             x ← Top(S_x)  //  Ho finito di lavorare con la x corrente, quindi prendo
                                quella precedente senza rimuoverla dallo stack
20)             if last = x.sx then  // Torno da sinistra
21)                 z_L ← ret          // Sto tornando dalla prima chiamata ricorsiva,
22)                 j ← Top(S_j)       // quindi inserisce il valore di ritorno in z_L
23)                 h_L ← Top(S_hL)    // e riprende i valori delle variabile pushate
24)                 (k_R, h_R) ← F_in(x, h_L, z_L)
25)                 if x.dx = ⊥ then  // Vedi note sotto per la spiegazione di questo if
26)                     z_R ← F_fin(F_⊥(F_ini(k_R, j)))
27)                     m ← F_post(x, h_L, z_L, h_R, z_R)
28)                     z ← F_fin(m)
29)                     ret ← z    // È stato fatto un return, in questo caso non imposto
30)                     last ← x  // call a false perché è gia false
31)                     (S_x, S_j, S_hL) ← (Pop(S_x), Pop(S_j), Pop(S_hL))
32)                 else
33)                     S_zL ← Push(S_zL, z_L)  // Preparo gli stack per fare la chiamata ricorsiva, il push di j
34)                     S_hR ← Push(S_hR, h_R)  // in questo caso è inutile perché il suo valore non cambia
35)                     (x, i) ← (x.dx, k_R)  //  Non pusho x perché prima è stato fatto il Top, non il Pop.
                                                  Inserirei lo stesso valore
36)                     call ← true  // È statà fatta una chiamata, quindi la funzione scende
37)             else // Torno da destra
38)                 z_R ← ret
39)                 (S_hL, h_L) ← Top&Pop(S_hL)
40)                 (S_hR, h_R) ← Top&Pop(S_hR)
41)                 (S_zL, z_L) ← Top&Pop(S_zL)
42)                 S_j ← Pop(S_j)
43)                 ret ← F_fin(F_post(x, h_L, z_L, h_R, z_R))
44)                 S_x ← Pop(S_x)
45)                 last ← x
46)     return ret
```