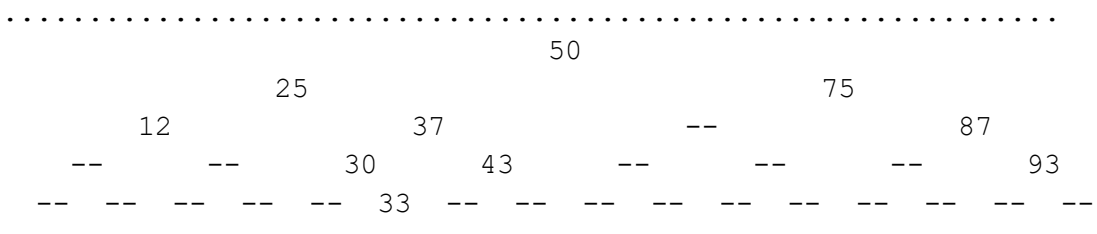
Napisz program w *Javie*, który dla zadanej listy węzłów (bez duplikatów), będących opisem przeglądu drzewa w porządku *PREORDER* lub *POSTORDER* wyznaczy **rekurencyjnie** drzewo BST a następnie wykona na nim następujące operacje:

1. PREORDER– wypisuje listę węzłów w porządku preorder
2. INORDER– wypisuje listę węzłów w porządku inorder
3. POSTORDER– wypisuje listę węzłów w porządku postorder
4. LEVELORDER – wypisuje listę węzłów w porządku levelorder
5. PARENT x – zwraca klucz ojca węzła o kluczu x
6. INSERT x - wstawia nowy węzeł o kluczu x, przy czym jeśli w drzewie element o kluczu x już występuje to go nie wstawia.
7. DELETE x - usuwa węzeł o kluczu x, przy czym w przypadku, gdy usuwany węzeł ma dwóch potomków, zamienia go losowo z jego następnikiem lub poprzednikiem.
8. SUCCESSOR x - zwraca klucz następnika węzła o kluczu x
9. PREDECESSOR x – zwraca klucz poprzednika węzła o kluczu x
10. DISPLAY- wypisuje drzewo, np. dla listy PREORDER postaci:
    PREORDER 50 25 12 37 30 33 43 75 87 93 drzewo wygląda jak poniżej



W przypadku braku węzła o kluczu x lub gdy węzeł o kluczu x nie ma ojca, następnika lub poprzednika program wypisze słowo ”BRAK”.

Wejście

Dane do programu wczytywane są ze standardowego wejścia (klawiatury) zgodnie z poniższą specyfikacją:

- Pierwsza linia zawiera liczbę całkowitą n ($1 \leq n \leq 10^6$), oznaczająca ilość wierzchołków drzewa binarnego.
- druga linia zawiera dokładnie jedno ze słów *PREORDER* lub *POSTORDER*.
- w kolejnej linii znajduje się n różnych kluczy (typu int) wypisanych w wyżej wymienionym porządku.
- w linii czwartej znajduje się liczba operacji m ($1 \leq m \leq 100$) do wykonania na utworzonym drzewie
- w każdej następnej linii znajduje się jedna z wymienionych wyżej operacji i ewentualnie jej argument.

Wyjście

Dla każdej operacji wypisz w jednej linii jej wynik zgodnie z podanymi przykładami. Ostatni węzeł na każdej z list kończy znak nowej linii.

Wymagania implementacyjne

Jedynym możliwym importem jest **java.util.Scanner**.

Uniwersytet Jagielloński Instytut Informatyki i Matematyki Komputerowej KIS Katedra Informatyki Stosowanej	METODY PROGRAMOWANIA 2018/2019 DRZEWA BST	P_09

Uwaga.

1. Klasa węzeł ma postać:

```
class Node   {
   public int info;       // element danych (klucz)
   public Node left;      // lewy potomek węzła
   public Node right;     // prawy lewy potomek węzła
   public Node(int info) {
        this.info  = info;
        this.left  = null;
        this.right = null;
   }
}  // koniec klasy Node
```

2. Wszystkie operacje oprócz DISPLAY muszą być **w wersji iteracyjnej** .
3. Operacje SUCCESSOR *x* i PREDECESSOR *x* mogą korzystać z operacji PARENT ale nie mogą korzystać z listy INORDER.

Przykład.

test.in:	**test.out:**
1 *10* *PREORDER* *50 25 12 37 30 33 43 75 87 93* *13* *POSTORDER* *LEVELORDER* *PARENT 33* *SUCCESSOR 50* *PREDECESSOR 50* *PARENT 50* *DELETE 50* *PARENT 50* *POSTORDER* *LEVELORDER* *INSERT 35* *INORDER* *LEVELORDER*	ZESTAW: 1 POSTORDER: 12 33 30 43 37 25 93 87 75 50 LEVELORDER: 50 25 75 12 37 87 30 43 93 33 PARENT 33: 30 SUCCESSOR 50: 75 PREDECESSOR 50: 43 PARENT 50: BRAK PARENT 50: BRAK POSTORDER: 12 33 30 43 37 25 93 87 75 LEVELORDER: 75 25 87 12 37 93 30 43 33 INORDER: 12 25 30 33 35 37 43 75 87 93 LEVELORDER: 75 25 87 12 37 93 30 43 33 35