Napisz program w *Javie*, który z danej listy węzłów niezawierających 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 – wyznacza klucz ojca węzła o kluczu x
6. INSERT x - wstawia nowy węzeł o kluczu x
7. DELETE x - usuwa węzeł o kluczu x, przy czym w przypadku, gdy usuwany węzeł ma dwóch potomków, zamienia go z jego następnikiem.
8. SUCCESSOR x - wyznacza klucz następnika węzła o kluczu x
9. PREDECESSOR x – wyznacza klucz poprzednika węzła o kluczu x

Przy czym w przypadku operacji INSERT x - jeśli w drzewie element o kluczu x już występuje to go nie wstawia. W pozostałych operacjach w przypadku braku węzła o kluczu x lub gdy dany węzeł x nie ma ojca lub 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 spacja.

Wymagania implementacyjne

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

Uwaga.

- 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;
        left = null;
        right = null;
    }
}  // koniec klasy Node
```

- Wszystkie operacje muszą być w wersji iteracyjnej (nie mogą zawierać rekurencji).
- Operacje SUCCESSOR *x* i PREDECESSOR *x* mogą korzystać z metody PARENT() ale nie mogą korzystać z listy INORDER.

Przykład.

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