class SegmentTree {

int st[]; // The array that stores segment tree nodes

public SegmentTree(int arr[], int n){

int x = (int) (Math.ceil(Math.log(n) / Math.log(2)));

int max\_size = 2 \* (int) Math.pow(2, x) - 1;

st = new int[max\_size]; // Memory allocation

constructSTUtil(arr, 0, n - 1, 0);

}

public int getMid(int s, int e) {return s + (e - s) / 2;}

public int constructSTUtil(int arr[], int ss, int se, int si){

if (ss == se) {st[si] = arr[ss]; return arr[ss];}

int mid = getMid(ss, se);

st[si] = constructSTUtil(arr, ss, mid, si \* 2 + 1) + constructSTUtil(arr, mid + 1, se, si \* 2 + 2);

return st[si];

}

public int getSum(int n, int qs, int qe){

if (qs < 0 || qe > n - 1 || qs > qe) {

System.out.println("Invalid Input");return -1;

}

return getSumUtil(0, n - 1, qs, qe, 0);

}

public int getSumUtil(int ss, int se, int qs, int qe, int si){

if (qs <= ss && qe >= se)return st[si];

if (se < qs || ss > qe)return 0;

int mid = getMid(ss, se);

return getSumUtil(ss, mid, qs, qe, 2 \* si + 1) +getSumUtil(mid + 1, se, qs, qe, 2 \* si + 2);

}

public void updateValue(int arr[], int n, int i, int new\_val){

if (i < 0 || i > n - 1) {

System.out.println("Invalid Input"); return;

}

int diff = new\_val - arr[i]; arr[i] = new\_val;

updateValueUtil(0, n - 1, i, diff, 0);

}

public void updateValueUtil(int ss, int se, int i, int diff, int si){

if (i < ss || i > se) return;

st[si] = st[si] + diff;

if (se != ss) {

int mid = getMid(ss, se);

updateValueUtil(ss, mid, i, diff, 2 \* si + 1);

updateValueUtil(mid + 1, se, i, diff, 2 \* si + 2);

}

}

public static void main(String args[]){

int arr[] = {1, 3, 5, 7, 9, 11};

int n = arr.length;

SegmentTree tree = new SegmentTree(arr, n);

System.out.println("Sum of values in given range = "+tree.getSum(n, 1, 3));

tree.updateValue(arr, n, 1, 10);

System.out.println("Updated sum of values in given range = " + tree.getSum(n, 1, 3));

}