```cpp
#include <bits/stdc++.h>

using i64 = long long;

struct FUNCTION {
    int n;

    std::vector<int> vis, c, d, prime;

    FUNCTION(int n) : n(n) {
        vis.resize(n);
        c.resize(n);
        d.resize(n);
        prime.resize(n);
    }

    void init() {
        vis[1] = 1;
        d[1] = 1;
        c[1] = 1;
        for (int i = 2; i <= n; i++) {
            if (!vis[i]) {
                prime.push_back(i);
                c[i] = i;
                d[i] = 2;
            }
            for (int j = 0; j < prime.size() && i * prime[j] <= n; j++) {
                vis[i * prime[j]] = 1;
                if (i % prime[j] == 0) {
                    c[i * prime[j]] = c[i] * prime[j];
                    d[i * prime[j]] = d[i] + d[i / c[i]];
                    break;
                }
                d[i * prime[j]] = d[i] * 2;
                c[i * prime[j]] = prime[j];
            }
        }
    }
};

#include <bits/stdc++.h>

using i64 = long long;

struct TREE {
    int n, flag = 0;

    std::vector<std::vector<int>> e, fa;

    std::vector<int> dep;

    TREE(int n) : n(n) {
        e.resize(n);
        fa.resize(n);
        dep.resize(n);
        for (int i = 0; i < n; i++) {
            fa[i].resize(21);
        }
    }

    void add(int u, int v) {
        e[u].push_back(v);
        e[v].push_back(u);
    }

    void dfs(int u, int last) {
        dep[u] = dep[last] + 1;
        for (int i = 1; i <= 20; i++) {
            if (!fa[u][i - 1]) break;
            fa[u][i] = fa[fa[u][i - 1]][i - 1];
        }
        for (auto v : e[u]) {
            if (v == fa[u][0]) continue;
            fa[v][0] = u;
            dfs(v, u);
        }
    }

    int lca(int x, int y) {
        for (int i = dep[x] - dep[y], j = 0; i > 0; i >>= 1, j++) {
            if (i & 1) x = fa[x][j];
        }
        for (int i = dep[y] - dep[x], j = 0; i > 0; i >>= 1, j++) {
            if (i & 1) y = fa[y][j];
        }
        if (x != y) {
            for (int i = 20; i >= 0; i--) {
                if (fa[x][i] == fa[y][i]) continue;
                x = fa[x][i];
                y = fa[y][i];
            }
            x = fa[x][0];
        }
        return x;
    };
};

#include <bits/stdc++.h>

using i64 = long long;

template<typename cmp>
struct DELHEAP {
    i64 sum = 0;

    std::priority_queue<int, std::vector<int>, cmp> q, d;

    int size() { return q.size() - d.size(); }

    bool empty() { return size() == 0; }
    void push(int x) { q.push(x); sum += x; }

    void erase(int x) { d.push(x); sum -= x; }

    int top() {
        while (!q.empty() && !d.empty() && q.top() == d.top()) {
            q.pop();
            d.pop();
        }
        return q.top();
    }

    void pop() { erase(q.top()); }
};

struct SET_MID {
    DELHEAP<std::less<int>> L;
    DELHEAP<std::greater<int>> R;

    void adjust() {
        while (L.size() > R.size()) {
            R.push(L.top());
            L.pop();
        }
        while (R.size() > L.size()) {
            L.push(R.top());
            R.pop();
        }
    }

    void insert(int x) {
        if (L.empty() || x <= L.top()) {
            L.push(x);
        } else {
            R.push(x);
        }
        adjust();
    }

    void erase(int x) {
        if (x <= L.top()) {
            L.erase(x);
        } else {
            R.erase(x);
        }
        adjust();
    }

    i64 sum() {
        i64 res = R.sum - L.sum;
        if (L.size() > R.size()) {
            res += L.top();
        }
        return res;
    }
```

```
};

#include <bits/stdc++.h>

using i64 = long long;

struct RMQ {
    int n;

    std::vector<std::array<int, 30>> fmin, fmax;

    RMQ(std::vector<int> a) {
        n = a.size() - 1;
        fmin.resize(n + 1);
        fmax.resize(n + 1);
        for (int i = 1; i <= n; i++) {
            fmin[i][0] = fmax[i][0] = a[i];
        }
        for (int i = 1; i < 30; i++) {
            if (1 << i > n) break;
            for (int j = 1; j + (1 << i) - 1 <= n; j++) {
                fmin[j][i] = std::min(fmin[j][i - 1], fmin[j + (1 << i - 1)][i - 
1]);
                fmax[j][i] = std::max(fmax[j][i - 1], fmax[j + (1 << i - 1)][i - 
1]);
            }
        }
    }

    int max(int l, int r) {
        int k = log(r - l + 1);
        return std::max(fmax[l][k], fmax[r - (1 << k) + 1][k]);
    }

    int min(int l, int r) {
        int k = log(r - l + 1);
        return std::min(fmin[l][k], fmin[r - (1 << k) + 1][k]);
    }
};
```