```cpp
#include <cstdio>
#include <vector>
using namespace std;

typedef vector<int> vi;

// Union-Find Disjoint Sets Library written in OOP manner, using both path
compression and union by rank heuristics
class UnionFind {                                               // OOP style
private:
  vi p, rank, setSize;                       // remember: vi is vector<int>
  int numSets;
public:
  UnionFind(int N) {
    setSize.assign(N, 1); numSets = N; rank.assign(N, 0);
    p.assign(N, 0); for (int i = 0; i < N; i++) p[i] = i; }
  int findSet(int i) { return (p[i] == i) ? i : (p[i] = findSet(p[i])); }
  bool isSameSet(int i, int j) { return findSet(i) == findSet(j); }
  void unionSet(int i, int j) {
    if (!isSameSet(i, j)) { numSets--;
    int x = findSet(i), y = findSet(j);
    // rank is used to keep the tree short
    if (rank[x] > rank[y]) { p[y] = x; setSize[x] += setSize[y]; }
    else                   { p[x] = y; setSize[y] += setSize[x];
                             if (rank[x] == rank[y]) rank[y]++; } } }
  int numDisjointSets() { return numSets; }
  int sizeOfSet(int i) { return setSize[findSet(i)]; }
};

int main() {
  printf("Assume that there are 5 disjoint sets initially\n");
  UnionFind UF(5); // create 5 disjoint sets
  printf("%d\n", UF.numDisjointSets()); // 5
  UF.unionSet(0, 1);
  printf("%d\n", UF.numDisjointSets()); // 4
  UF.unionSet(2, 3);
  printf("%d\n", UF.numDisjointSets()); // 3
  UF.unionSet(4, 3);
  printf("%d\n", UF.numDisjointSets()); // 2
  printf("isSameSet(0, 3) = %d\n", UF.isSameSet(0, 3)); // will return 0 (false)
  printf("isSameSet(4, 3) = %d\n", UF.isSameSet(4, 3)); // will return 1 (true)
  for (int i = 0; i < 5; i++) // findSet will return 1 for {0, 1} and 3 for {2, 3,
4}
    printf("findSet(%d) = %d, sizeOfSet(%d) = %d\n", i, UF.findSet(i), i,
UF.sizeOfSet(i));
  UF.unionSet(0, 3);
  printf("%d\n", UF.numDisjointSets()); // 1
  for (int i = 0; i < 5; i++) // findSet will return 3 for {0, 1, 2, 3, 4}
    printf("findSet(%d) = %d, sizeOfSet(%d) = %d\n", i, UF.findSet(i), i,
UF.sizeOfSet(i));
  return 0;
}
```