```cpp
#include <iostream>
#include <string>
#include <vector>
#include <unordered_set>

using namespace std;

// For this assignment, when a value is deleted, ALL occurrences are removed from
// the bag. There is NO condition for adding values or on the number of repeated
// values. It's NOT important to retrieve actual values, it's just important to
// check whether AT LEAST 1 occurrence of a value is in the bag or not.

// For this kind of exercise a set is well suited.


int main() {
    std::string command;
    std::unordered_set<int> Bag;
    int x;

    std::cin >> command;

    while (command != "quit") {
        if (command == "add") {
            std::cin >> x;
            Bag.insert(x);
        }

        else if (command == "del") {
            std::cin >> x;
            // If the bag is not empty...
            // ...and If x is in the Bag, it's deleted
            if (!Bag.empty())
                Bag.erase(x);
        }

        else if (command == "qry") {
            std::cin >> x;
            if (Bag.empty())                    // Nothing to extract
                std::cout << "F";
            // If 'find' doesn't find x, it will reach the end of Bag
            else if (Bag.find(x) != Bag.end())
                std::cout << "T";
            else
                std::cout << "F";
        }

        else {
            std::cout << "error!" << endl;
            return 0;
        }

        std::cin >> command;
    }
    return 0;
}


```

```

/*
The problem can be solved with vectors but in a vector repeated values take
(contigous) memory space, but multiple occurrences of a value are not needed.

A set prevents repeated values (it doesn't do anything when calling the method
'insert' with a value that is already in the set).
Search should be faster for sets compared to vectors and Unordered sets,
specifically, should be even faster.

sets don't provide random access with indeces, but it's not needed.
*/
```