```cpp
#include <iostream>
#include <string.h>
#include <unordered_set>

using namespace std;


// In this case repeated values in the bag ARE IMPORTANT.
// A multiset allows for repeated values storage.
// Unordered multisets are faster at accessing values than both
// multisets and vectors.

// The code is almost unchanged compared to the previous assignment.

// By this time in the course I'm not supposed to know templates but I
// used them anyway to repeat code for int and double numbers since
// code is exactly the same in both cases.


template <typename T>
void funWithBags(unordered_multiset<T>);

int main() {
    string select;
    cout << "Choose int or double: ";
    cin >> select;

    // Compiler doesn't like to have the function call outside.
    // Potentially this if-else may not execute anything and
    // Bag wouldn't be declared before the function call.
    if (select == "int") {
        unordered_multiset<int> Bag;          // Assignment 4-2
        funWithBags(Bag);
    }
    else if (select == "double") {
        unordered_multiset<double> Bag;       // Assignment 4-3
        funWithBags(Bag);
    }
    else
        cout << "Error: Nothing valid selected";
    return 0;
}
```

```cpp
template <typename T>
void funWithBags(unordered_multiset<T> Bag) {
    string command;
    T x;

    cin >> command;

    while (command != "quit") {
        if (command == "add") {
            cin >> x;
            Bag.insert(x);
        }
        else if (command == "del") {
            cin >> x;
            // The only difference compared to the previous
            // assignment is here (read below) *:
            if (!Bag.empty())
                Bag.erase(Bag.find(x));
        }
        else if (command == "qry") {
            cin >> x;
            if (Bag.empty())
                cout << "F";
            else if (Bag.find(x) == Bag.end())
                cout << "F";
            else
                cout << "T";
        }
        else {
            cout << "error!" << endl;
            return;
        }
        cin >> command;
    }
}

// * Using the method 'erase' with a value like in the
// previous assignment will remove EVERY occurrences.
// I have to use the method 'find' to get the position of one
// element and pass the position to 'erase'.

// Actually exactly the same code can be used for all point in the
// assignment but for the first point, it would be superfluous to
// give an iterator to the method erase.

// A vector now may be used more easily because of the repeated values
// but vectors are slower at retrieving values.

// In the course, an automatic online testing system was used to hand in
// the assignments and this source code couldn't pass because the choice
// between the type of elements was not expected by the test...
```