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

using namespace std;

void display(std::vector<int>::iterator, std::vector<int>::iterator);
void interDisplay(std::vector<int>&, std::vector<int>&);
bool createSet(const std::string&, std::vector<int>&, std::vector <int>&);



// The main function is the same as the previous assignment
// without the sorting of the 2 datasets.

int main() {
    std::string input;
    std::vector<int> SetA, SetB;

    std::getline(cin, input);
    if (createSet(input, SetA, SetB)) {
        interDisplay(SetA, SetB);
    }
    else
        std::cout << "Error" << std::endl;
    return 0;
}



///////////////////////////////////////////////////////////////////////



// This function is also unchanged

bool createSet(const string & input, vector<int> & SetA, vector <int> & SetB) {
    int n;
    char c;
    std::stringstream stream(input);

    while (stream >> c) {
        if (c == 'a') {
            stream >> n;
            SetA.push_back(n);
        }
        else if (c == 'b') {
            stream >> n;
            SetB.push_back(n);
        }
        else
            return false;
    }
    return true;
}


```

```cpp
////////////////////////////////////////////////////////////////////////
// Compared to the first assignment, now data point have to be printed
// alternatively. This creates a complication if the 2 datasets have
// different sizes.
////////////////////////////////////////////////////////////////////////


void interDisplay(std::vector<int>& Va, std::vector<int>& Vb) {
    // Get the sizes once to avoid recalling the method size multiple times
    unsigned lA = Va.size();
    unsigned lB = Vb.size();

    // Use a different name (a reference) for the shorter size
    unsigned& shorter = (lA <= lB) ? lA : lB;

    // print alternatively until this size
    for (auto i = 0 ; i < shorter ; i++)
        std::cout << Va[i] << " " << Vb[i] << " ";

    // Print the additional elements of the longer dataset.
    // Delegate the printing to another function using iterators for a class
    // vector
    if (lA > lB)
        display(Va.begin() + shorter - 1, Va.end());
    else if (lB > lA)
        display(Vb.begin() + shorter - 1, Vb.end());
    else
        return;
}










//////////////////////////////////////////////////////////////////////////////
// Takes 2 iterators to address the excess of the longer vector.
//////////////////////////////////////////////////////////////////////////////


void display(std::vector<int>::iterator pos, std::vector<int>::iterator end) {
    for ( ; pos != end ; pos++)
        std::cout << *pos << " ";
}
```