```cpp
#include <iostream>
#include <list>
#include <vector>
using namespace std;



template <class T> ostream& operator<<(ostream& str, const vector<T>& V);
template <class T> ostream& operator<<(ostream& str, const vector<T*>& V);
template <class T> ostream& operator<<(ostream& str, const list<T>& L);
template <class T> ostream& operator<<(ostream& str, const list<T*>& L);
void DB1_to_DB2(vector<list<int>>& DB1, vector<list<int*>*>& DB2);
void DB2_to_DB3(vector<list<int*>*>& DB2, list<vector<int>*>& DB3);







int main() {

    vector<list<int>> DB1{ {1,2,3}, {4,5}, {6,7,8,9} };

    vector<list<int*>*> DB2;
    list<vector<int>*> DB3;

    DB1_to_DB2(DB1, DB2);
    cout << "DB1: " << DB1 << endl;
    cout << "DB2: " << DB2 << endl;

    DB2_to_DB3(DB2, DB3);
    cout << "DB2: " << DB2 << endl;
    cout << "DB3: " << DB3 << endl;



    system("pause");
    return 0;
}


void DB2_to_DB3(vector<list<int*>*>& DB2, list<vector<int>*>& DB3) {
    // 1. delete the current DB3
    for (auto it : DB3) {
        it->clear();
        delete it;
    }
    DB3.clear();
    // 2.loop build new DB3 with same value as DB2 (loop in DB2)
    for (auto it : DB2) {
        vector<int>* pv{ new vector<int> };
        for (auto itv : *it) {
                pv->push_back(*itv);
            }
            DB3.push_back(pv);
        }
}




void DB1_to_DB2(vector<list<int>>& DB1, vector<list<int*>*>& DB2) { // call by 
  reference '&' !!!!
    // 1. delete the current DB2
    for (auto it : DB2) {
        for (auto iti : *it) {
            delete iti;
        }
        it->clear();
        delete it;
    }
    DB2.clear();

    // 2.loop build new DB2 with same value as DB1 (loop in DB1)
    for (auto it : DB1) {
        list<int*>* pl{ new list<int*> }; // synchronously building DB2's layer 
          element
        for (auto itl : it) {
            int* pi{ new int(itl) };
            pl->push_back(pi);
        }
        DB2.push_back(pl);
    }
}






template <class T> ostream& operator<<(ostream& str, const vector<T>& V) {
    str << "[vector: ";
    for (auto& i : V) { str << i << " "; }
    str << "]";
    return str;
}

template <class T> ostream& operator<<(ostream& str, const vector<T*>& V) {

    str << "[vector: ";
    for (auto& i : V) { str << *i << " "; }
    str << "]";
    return str;
}
```

```

template <class T> ostream& operator<<(ostream& str, const list<T>& L) {

    str << "<list: ";
    for (auto& i : L) { str << i << " "; }
    str << ">";
    return str;

}

template <class T> ostream& operator<<(ostream& str, const list<T*>& L) {

    str << "<list: ";
    for (auto& i : L) { str << *i << " "; }
    str << ">";
    return str;

}
```