```
//HW7  Due: Tuesday, December 12.
/*
implementclass class bag, as desbribed in the class description.
Also implement the needed overloaded operator << for printing.  (See main function and
sample screenshot.)

No extra functions or external structures (such as array, list, vector, etc.) allowed.

*/

#include <iostream>
#include <vector>

using namespace std;


class ThreeD {//you are not allowed to modify this class.
public:
        int ht;
        int wid;
        int dep;
        ThreeD(int i, int j, int k) : ht(i), wid(j), dep(k) {}
        ThreeD() { ht = wid = dep = 0; }
        int vol() const { return ht * wid * dep; }

};

ostream& operator<<(ostream& str, const ThreeD& t) { str << "[" << t.ht << ", " << t.wid <<
", " << t.dep << "]"; return str; }

template <class T1> class node {//you are not allowed to modify this class
public:
        T1 value;
        node<T1>* next = nullptr;
        node<T1>* previous = nullptr;
        node<T1>(T1 t) { next = previous = nullptr; value = t; }
};

template <class T1, class T2 = less<T1>> class bag {
public:
        //There is a doubly linked list structre inside bag.
        node<T1>* head = nullptr;
        node<T1>* tail = nullptr;
        bag() { head = tail = nullptr; }

        //Implement the following 5 functions including Sort.

        bag(const initializer_list<T1>& I);//Initializer List;
        bag(const bag<T1, T2>& B);  //Copy Constructor;
        void operator=(const bag<T1, T2>& B); //Copy Assignment
        ~bag<T1, T2>();//Destructor

private:

        void Sort(T2 func);//Only to be invoked by other member functions

        /*Sort will implement bubble sort to create an ascending(i.e., increasing) sequence
        and will remove duplicates.
        T2 will be the functorClass (similar to the way map and set do for sorting).
        */
};

class myFunctorClass {
public:
 //Implement all needed functors.  See the main functions
};

template <typename T1, typename T2 = less<T1>> ostream& operator<<(ostream& str, const 
bag<T1, T2>& t); //forward printing
template <typename T1, typename T2 = less<T1>> ostream& operator<<(ostream& str, const 
bag<T1, T2>&& t); //Reverse printing
template <typename T1> ostream& operator<<(ostream& str, const vector < T1*> v);
template <typename T1> ostream& operator<<(ostream& str, const vector < T1> v);
int main() {
        bag<int> B0{ 3,4,1,4,5,6, 2, 1 };
        cout << B0 << endl;
        auto B0a{ B0 };
        cout << move(B0a) << endl;

        bag<int> B1{ B0 };
        auto B1a{ B1 };
        cout<<endl << B1 << endl;
        cout << move(B1a) << endl;
        bag<int> B2{ B0 };
        B2 = B1;
        auto B2a{ B2 };
        cout<<endl << B2 << endl;
        cout << move(B2) << endl;

        bag<vector<int*>, myFunctorClass> B3{ {new int {5}, new int{7}, new int{11}}, {new 
int {2}, new int {9}} };
        auto B3a{ B3 };
        cout <<endl<< B3 << endl;
        cout << move(B3a) << endl;


        ThreeD t1{ 1,3,2 }, t2{ 4,0,0 }, t3{ 3, 1,1 };

        bag<ThreeD, myFunctorClass> B4{ t1, t2, t3 };
        auto B4a{ B4 };
        cout <<endl<< B4 << endl;
        cout << move(B4a) << endl;

        vector<bag<int>> V1{ {1,2,5}, {3,1,1}, {0,1,2} };
        cout <<endl<< V1 << endl;
        bag<int> b11{ 5,4,3, 5 }, b22{ 4,3,2, 3 }, b33{ 1, 2,3,2 };

        auto p2 = new bag<int>{ 1,2,3,1 };
        cout << endl<<*p2 << endl;
        delete p2;
        auto p = new node<bag<int>>{ b11 };
        cout <<endl<< p->value << endl;
        delete p;
        bag<bag<int>, myFunctorClass> B6{ {  5,4,3, 5}, {4,3,2, 3}, {1, 2,3,2} };
        auto B6a{ B6 };
        cout <<endl<< B6 << endl;
        cout << move(B6a) << endl;
        bag<bag<int>, myFunctorClass> B7{ b11, b22, b33 };
        auto B7a{ B7 };
        cout <<endl<< B7 << endl;
        cout << move(B7a) << endl;


        auto p10{ new node<vector<bag<int>>>   {    {{9, 4, 2}, { 3,3,3 }, { 1,2,3 }}     } 
};
        cout <<endl<< p10->value << endl;
        delete p10;
        bag<vector<bag<int>>, myFunctorClass> B8{ {{9,4, 2}, {3,3,3}, {1,2,3}}, {{4,1}, 
{0,5, 7}}, {{2,1,2}, {0,1,2}, {0,2,1,0}} };
```

```
        auto B8a{ B8 };
        cout <<endl<< B8 << endl;
        cout << move(B8a) << endl;
        return 0;
}
```