```
//HW6.  Due: Tuesday, Nov. 28 at 11:59pm

#include <iostream>
#include <map>
#include <list>
#include <vector>
#include <set>
#include <unordered_set>
#include <functional>

using namespace std;


class node {
public:
        int value;
        node* next;
        node(int i) { value = i; next = nullptr; }//construct
        node() { next = nullptr; }//default constructor

};

class LinkedList {
public:
        node* head;
        LinkedList(int n, int m);//Constructor for an n-node linked list
        //with values randomly in 0 ... m-1


        LinkedList() { head = nullptr; } //default constructor

        void Sort();//You need to change this to allow the Sort examples in the main
functions.
        //You can re-use code from lecture 2023_09_07 .

};




//When sorting structures, always sort the sums of all numbers within the structure.
//when comparing equality of two structures, always compare sums of all int values in the
structures
//When Hashing a structure, always hash the sum of all int values in the structure
//For printing, following the following natations.  map: { ...}   set(or unordered set):
<...>




int main() {
        //The following statement won't compile. Fix it.
        //Set initial values for S1.  Every structure, set or map, needs to have at least 3
elements.

        set<map < map <int*, string >*, set <int>*>> S1;

        //You need to support the following statement.
        cout << S1 << endl;


        //The following statement won't compile. Fix it.
        //Set initial values for H1.  Every structure, set or map, needs to have at least 3
```

```
elements.
        unordered_set<map < map <int*, string >*, set <int>*  >> H1;

        //You need to support the following statement.
        cout << H1 << endl; //Print bucket.  See sample screenshots.



        LinkedList L1{ 20, 10 };
        cout << L1 << endl;
        L1.Sort();
        cout << L1 << endl;

        L1.Sort([](int a, int b) {return a % 3 < b % 3; });
        cout << L1 << endl;
        L1.Sort(greater<int>{});
        cout << L1 << endl;


        return 0;
}
```