#include <iostream>

#include <string>

class hashtable {

public:

struct item {

int key = 0;

std::string data = "";

};

hashtable(int size) {

this->capacity = size;

items = new item \* [size];

for (int i = 0; i < size; i++) {

items[i] = nullptr;

}

}

~hashtable() {

for (int i = 0; i < capacity; i++) {

if (items[i] != nullptr)

delete items[i];

}

delete[]items;

}

void setData() {

int key;

std::string data;

std::cin >> key >> data;

int index = getFirstFreeIndex(key);

items[index] = new item;

items[index]->key = key;

items[index]->data = data;

}

void printData() {

for (int i = 0; i < capacity; i++) {

if (items[i] != nullptr) {

std::cout << i << " " << items[i]->key << " " << items[i]->data << std::endl;

}

}

}

void deleteItem(long key) {

int h = hash(key);

if (items[h]->key == key) {

delete items[h];

items[h] = nullptr;

}

else {

for (int i = 0; i < capacity; i++) {

int newIndex = (i + h + 1) % capacity;

if (items[newIndex] != nullptr && items[newIndex]->key == key) {

delete items[newIndex];

items[newIndex] = nullptr;

break;

}

}

}

}

private:

int getFirstFreeIndex(long key) {

int index = hash(key);

if (items[index] != nullptr) {

for (int i = 0; i < capacity; i++) {

int newIndex = (i + index + 1) % capacity;

if (items[newIndex] == nullptr) {

index = newIndex;

break;

}

}

}

return index;

}

int hash(long key) {

return key % 10;

}

int capacity;

item\*\* items;

};

int main()

{

std::string order;

int lp = 0;

long key2 = 0;

std::cout << "Liczba przypadkow ";

std::cin >> lp;

for (int i = 0; i < lp; i++) {

std::cin >> order;

std::cin >> key2;

hashtable hashtab(key2);

do {

std::cin >> order;

if (order == "add") {

hashtab.setData();

}

else if (order == "print") {

hashtab.printData();

}

else if (order == "delete") {

std::cin >> key2;

hashtab.deleteItem(key2);

}

} while (order != "stop");

}

system("PAUSE");

}