

import java.util.Scanner;

class SLLNode {

String name;

int price;

SLLNode succ;

public SLLNode(String name, int price, SLLNode succ) {

this.name = name;

this.price = price;

this.succ = succ;

}

@Override

public String toString() {

return name;

}

}

class SLL {

SLLNode first;

public SLL() {

this.first = null;

}

public void insertFirst(String name, int price) {

SLLNode ins = new SLLNode(name, price, first);

first = ins;

}

public void insertLast(String name, int price) {

if (first != null) {

SLLNode tmp = first;

while (tmp.succ != null)

tmp = tmp.succ;

SLLNode ins = new SLLNode(name, price, null);

tmp.succ = ins;

} else {

insertFirst(name, price);

}

}

@Override

public String toString() {

StringBuilder ret = new StringBuilder();

if (first != null) {

SLLNode tmp = first;

ret.append(tmp).append("\n");

while (tmp.succ != null) {

tmp = tmp.succ;

ret.append(tmp).append("\n");

}

} else

ret = new StringBuilder("NO ELEMENTS");

return ret.toString();

}

}

public class CakeShop {

public static void removeCakes(SLL cakes) {

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int n = Integer.parseInt(scanner.nextLine());

SLL cakes = new SLL();

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

String line = scanner.nextLine();

String[] parts = line.split("\\s+");

cakes.insertLast(parts[0], Integer.parseInt(parts[1]));

}

removeCakes(cakes);

System.out.println(cakes.toString());

}

}

3

Chocolate-Cake 100

Vanila-Cake 100

Carrot-Cake 101

Izlez:

Chocolate-Cake

Vanila-Cake

4

Chocolate-Cake 100

Vanila-Cake 100

Carrot-Cake 101

Cheese-Cake 101

Izlez:

Chocolate-Cake

Vanila-Cake

4

Cheese-Cake 101

Chocolate-Cake 100

Vanila-Cake 100

Carrot-Cake 101

Izlez:

Chocolate-Cake

Vanila-Cake

2

Cheese-Cake 100

Chocolate-Cake 101

Izlez:

Cheese-Cake

7

Carrot-Cake 800

Cheese-Cake 101

Chocolate-Cake 100

Vanilla-Cake 500

Pound-Cake 600

Red-Velvet-Cake 200

Vanilla-Cake 500

Izlez:

Cheese-Cake

Chocolate-Cake

Red-Velvet-Cake

3

Carrot-Cake 100

Cheese-Cake 101

Chocolate-Cake 100

Izlez:

Carrot-Cake

Chocolate-Cake

2

Carrot-Cake 101

Cheese-Cake 100

Izlez:

Cheese-Cake

7

Carrot-Cake 100

Cheese-Cake 200

Chocolate-Cake 300

Vanilla-Cake 100

Pound-Cake 200

Red-Velvet-Cake 200

Vanilla-Cake 300

Izlez:

Carrot-Cake

Cheese-Cake

Vanilla-Cake

Pound-Cake

Red-Velvet-Cake

**Inicijalen kod so klasa:**

import java.util.Iterator;

import java.util.NoSuchElementException;

import java.util.Scanner;

class SLLNode<E> {

protected E element;

protected SLLNode<E> succ;

public SLLNode(E elem, SLLNode<E> succ) {

this.element = elem;

this.succ = succ;

}

@Override

public String toString() {

return element.toString();

}

}

class SLL<E> {

private SLLNode<E> first;

public SLL() {

// Construct an empty SLL

this.first = null;

}

public void deleteList() {

first = null;

}

public int length() {

int ret;

if (first != null) {

SLLNode<E> tmp = first;

ret = 1;

while (tmp.succ != null) {

tmp = tmp.succ;

ret++;

}

return ret;

} else

return 0;

}

@Override

public String toString() {

String ret = new String();

if (first != null) {

SLLNode<E> tmp = first;

ret += tmp + "->";

while (tmp.succ != null) {

tmp = tmp.succ;

ret += tmp + "->";

}

} else

ret = "Prazna lista!!!";

return ret;

}

public void insertFirst(E o) {

SLLNode<E> ins = new SLLNode<E>(o, first);

first = ins;

}

public void insertAfter(E o, SLLNode<E> node) {

if (node != null) {

SLLNode<E> ins = new SLLNode<E>(o, node.succ);

node.succ = ins;

} else {

System.out.println("Dadenot jazol e null");

}

}

public void insertBefore(E o, SLLNode<E> before) {

if (first != null) {

SLLNode<E> tmp = first;

if(first==before){

this.insertFirst(o);

return;

}

//ako first!=before

while (tmp.succ != before)

tmp = tmp.succ;

if (tmp.succ == before) {

SLLNode<E> ins = new SLLNode<E>(o, before);

tmp.succ = ins;

} else {

System.out.println("Elementot ne postoi vo listata");

}

} else {

System.out.println("Listata e prazna");

}

}

public void insertLast(E o) {

if (first != null) {

SLLNode<E> tmp = first;

while (tmp.succ != null)

tmp = tmp.succ;

SLLNode<E> ins = new SLLNode<E>(o, null);

tmp.succ = ins;

} else {

insertFirst(o);

}

}

public E deleteFirst() {

if (first != null) {

SLLNode<E> tmp = first;

first = first.succ;

return tmp.element;

} else {

System.out.println("Listata e prazna");

return null;

}

}

public E delete(SLLNode<E> node) {

if (first != null) {

SLLNode<E> tmp = first;

if(first ==node){

return this.deleteFirst();

}

while (tmp.succ != node && tmp.succ.succ != null)

tmp = tmp.succ;

if (tmp.succ == node) {

tmp.succ = tmp.succ.succ;

return node.element;

} else {

System.out.println("Elementot ne postoi vo listata");

return null;

}

} else {

System.out.println("Listata e prazna");

return null;

}

}

public SLLNode<E> getFirst() {

return first;

}

public SLLNode<E> find(E o) {

if (first != null) {

SLLNode<E> tmp = first;

while (tmp.element != o && tmp.succ != null)

tmp = tmp.succ;

if (tmp.element == o) {

return tmp;

} else {

System.out.println("Elementot ne postoi vo listata");

}

} else {

System.out.println("Listata e prazna");

}

return first;

}

public Iterator<E> iterator () {

// Return an iterator that visits all elements of this list, in left-to-right order.

return new LRIterator<E>();

}

// //////////Inner class ////////////

private class LRIterator<E> implements Iterator<E> {

private SLLNode<E> place, curr;

private LRIterator() {

place = (SLLNode<E>) first;

curr = null;

}

public boolean hasNext() {

return (place != null);

}

public E next() {

if (place == null)

throw new NoSuchElementException();

E nextElem = place.element;

curr = place;

place = place.succ;

return nextElem;

}

public void remove() {

//Not implemented

}

}

public void mirror(){

if (first != null) {

//m=nextsucc, p=tmp,q=next

SLLNode<E> tmp = first;

SLLNode<E> newsucc = null;

SLLNode<E> next;

while(tmp != null){

next = tmp.succ;

tmp.succ = newsucc;

newsucc = tmp;

tmp = next;

}

first = newsucc;

}

}

public void merge (SLL<E> in){

if (first != null) {

SLLNode<E> tmp = first;

while(tmp.succ != null)

tmp = tmp.succ;

tmp.succ = in.getFirst();

}

else{

first = in.getFirst();

}

}

}

class Cake

{

String name;

int price;

public Cake(String name, int price)

{

this.name=name;

this.price=price;

}

public String toString()

{

return name+" "+price;

}

}

public class CakeShop {

public static void removeCakes(SLL<Cake> cakes) {

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int n = Integer.parseInt(scanner.nextLine());

SLL<Cake> cakes = new SLL<Cake>();

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

String line = scanner.nextLine();

String[] parts = line.split("\\s+");

Cake c=new Cake(parts[0], Integer.parseInt(parts[1]));

cakes.insertLast(c);

}

removeCakes(cakes);

System.out.println(cakes.toString());

}

}