CS 143 Assignment 1
Warehouse Class
See Canvas for due date!

You work for a company that has warehouses of items. The items are all packaged in pallets that are roughly the same size. Each warehouse is built to hold a certain maximum number of pallets. To keep a warehouse from holding too much of any one item, each warehouse also has a limit on how many pallets of each item it will store at a time. (If the warehouse can't store an item, that item would go to a different warehouse to make sure that items are evenly distributed, but your code won't track multiple warehouses.)

You will create a `Warehouse` class to simulate a warehouse according to the above description. If you would like to make use of another class of your design, declare it inside the Warehouse class (this is called an inner class). It is possible to create an efficient solution without having to do this. You can imagine the pallets sitting in a long line; what kind of data structure have we seen that can take advantage of this?

The constructor will be given the size (maximum number of pallets it can hold) and the limit for how many pallets of each item may be stored at a time. When the warehouse is constructed, it is empty. The get methods will report back the size and limit per item that the warehouse was constructed with. The constructor and get method declarations are defined below:

```
public Warehouse(int size, int limitPerItem)
```

```
public int getSize()
```

```
public int getLimitPerItem()
```

Your warehouse object will have a method to receive a truckload. A truckload will have only one kind of item (represented by an item code number) as well as the number of pallets of that item. You will load as many pallets as you can of that item given the above rules and return the number of pallets left on the truck (not loaded).

```
public int receive(int itemCode, int itemCount)
```

Your warehouse object will have a method to ship an item. The shipment request will contain the item code as well as the number of pallets requested. The warehouse will ship out as many as it can, up to the amount requested. Your method will return the number of pallets shipped of that item.

```
public int ship(int itemCode, int itemCount)
```

Both the `receive` and `ship` methods will have to update the internal tracking of items as items are received and shipped out. Different warehouses operate independently of each other and can have similar or entirely different sets of items.

You may assume that every integer (size, limit per item, item code, and item count) given to your methods is greater than 0.

The back side of this sheet describes an example scenario.

Suppose you create Warehouse A with room for 3 pallets but only allowing 2 of any item:
```
Warehouse A = new Warehouse(3, 2);
```

Warehouse A then receives 3 pallets of item 1. You can only hold 2 of any item, so your method returns 1 to say that it left one behind:
```
A.receive(1, 3) returns 1
```

Warehouse A receives 1 pallet of item 2. Your method returns 0 to say it left nothing behind:
```
A.receive(2, 1) returns 0
```

Warehouse A is requested to ship 1 pallet of item 1. Your method returns 1 to say that it shipped that one pallet:
```
A.ship(1, 1) returns 1
```

Warehouse A receives 2 pallets of item 1. Your method returns 1, since it leaves 1 behind after fitting 1 (which happens to be under the limit).

Warehouse A receives 1 pallet of item 4. Your method returns 1 – it has no room so it leaves that one behind.

Warehouse A is requested to ship 2 pallets of item 1. Your method returns 2 to say it shipped the last 2 pallets of the item requested.

Warehouse A receives 2 pallets of item 2. Your method returns 1, since it already had one, so it only stores one more.