Buy a Bag; Leave a Bag

You manage a “charity” organization that collects bags of goods and sells them to others. The profits go to constructing malls for underprivileged children. You have typically had to keep track of **all the locations** using pen and paper, but after taking CS I you feel confident that you can write a program to manage all this stuff automatically.

You work typically with 2 people. Some customers will drop off several bags. The other customers will buy as many bags as they can both afford and carry.

All you need to do is determine is the contents bought and the total profit from each customer (easy right?).

The only caveat is that due to the construction of each drop off location only the most recently place bag will be available. That is if bag 1 is dropped off and some time later bag 2 is dropped off, then bag 2 must be picked up (and subsequently purchased) prior to bag 1 (even if the customer can’t afford bag 2 but can afford bag 1).

Input Specification

The beginning of each update be an integer (either -1, 0, or 1).

When an update begins with -1, it means the customer is buying bags. Following “-1”, on the next line, will be three positive integers, ***k***, ***m,*** and ***c***, (***k*** < 1,000; ***m***, ***c*** < 100,000) representing the drop off location, the amount of money, and carrying capacity for the current customer. The customer will buy as many bags as possible. Output will be generated by the customer

When an update begins with a 1, it means the customer is dropping off bags. Following this “1” on the next line will be two positive integers, ***k*** and ***n***, (***k*** < 1,000) representing the drop off location and the number of bags the current customer is dropping off. The following ***n*** will be the description of the dropped off bags in the order they are placed into the drop off location. Each bag description will be on its own line. The bag description will begin with two integers, ***m*** and ***w***, representing the cost and the weight respectfully of the current bag. The bag description is finished with a single string of at most 19 Latin lowercase characters.

When an update begins with a 0, it means the input has terminated.

Output Specification

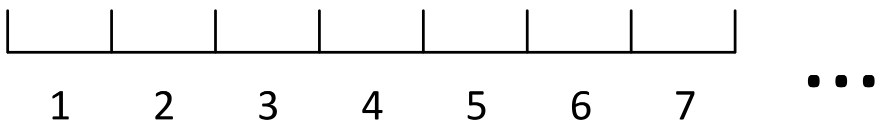
For each customer of type -1 output one line. The beginning of the line should contain a single integer ***p*** representing the total cost of the items sold. The rest of the line should be the string of the sold bags in the order they were purchased, each separated by a single space.

Input Output Example

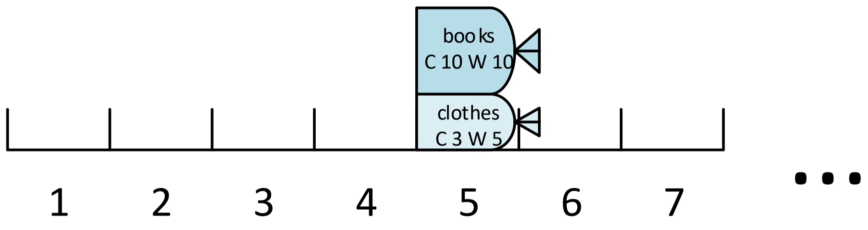
Input	Output
1 5 2 3 5 clothes 10 10 books 1 3 4 1 1 candy 19 1 movies 1 23 rocks 3 4 clothes -1 3 50 15 -1 3 47 15 -1 5 100 100 1 3 1 5 5 silverware 0	3 clothes 0 13 books clothes
-1 2 100 24 1 3 4 1 1 candy 19 1 movies 1 23 rocks 3 4 clothes -1 3 100 24 -1 3 97 24 -1 3 77 24 -1 3 74 24 0	0 3 clothes 20 rocks movies 1 candy 0

Explanation

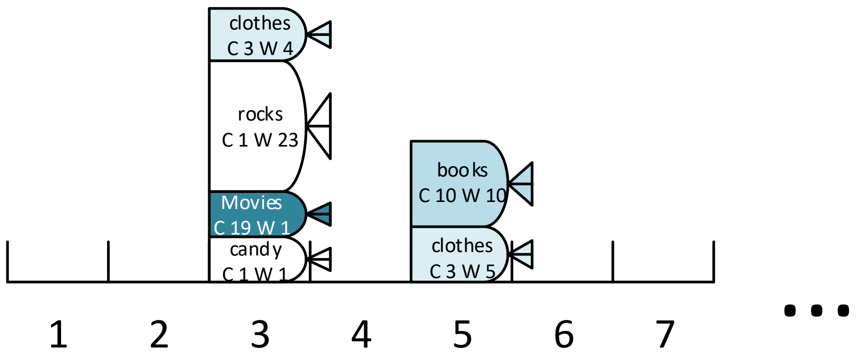
Case 1
Initially we have no bags



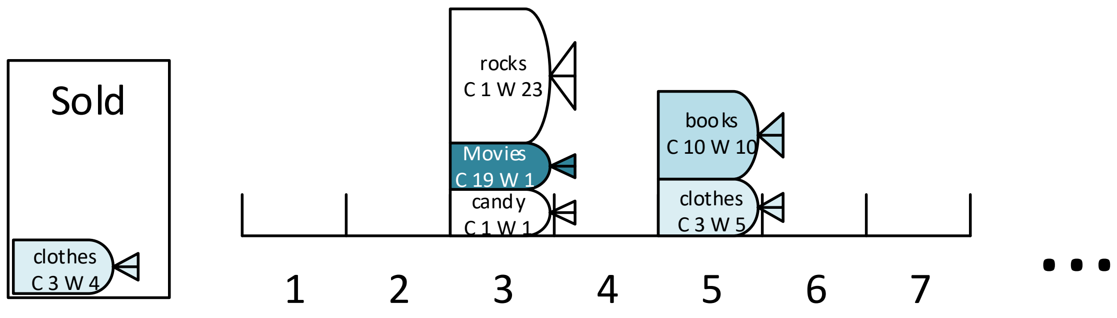
The first customer drops off two bags. We have no output.



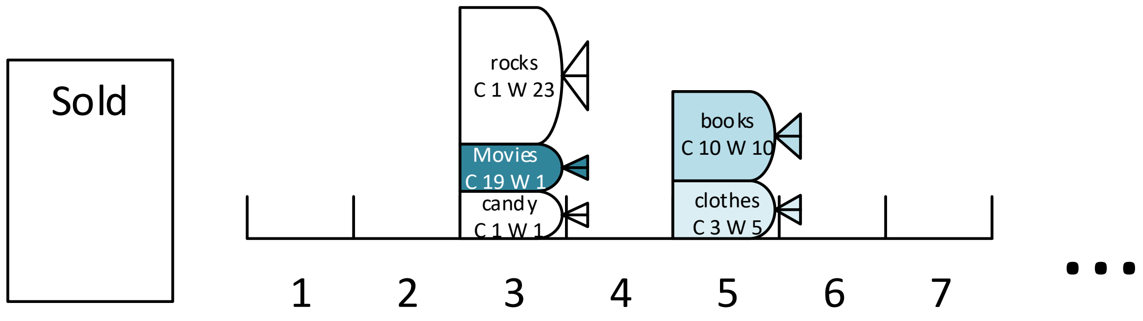
The second person drops of 2 bags at a different location.



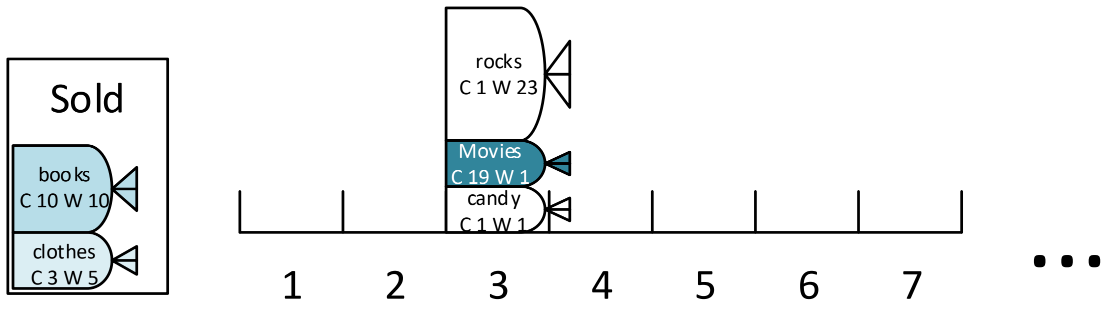
The third person buys a bag from location 3, but does not have enough carrying capacity to buy more bags.



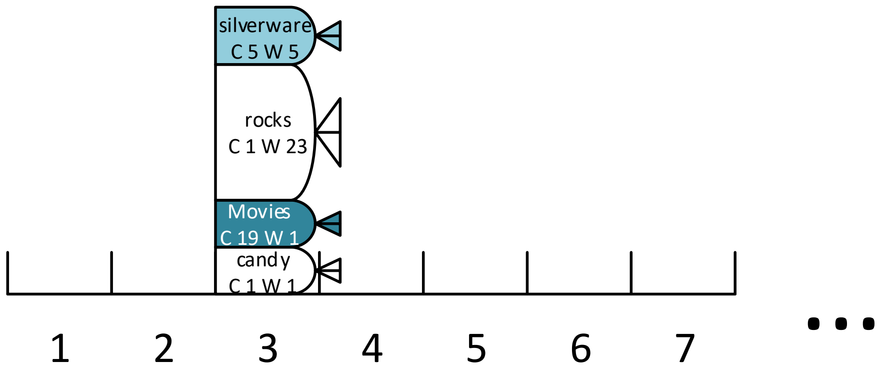
The next person comes to location 3 to buy things, but they don't have enough carrying capacity to buy anything.



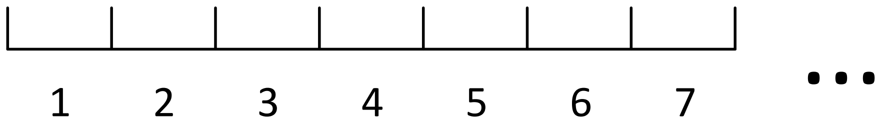
The fifth person goes to location 5 and buys all the bags there.



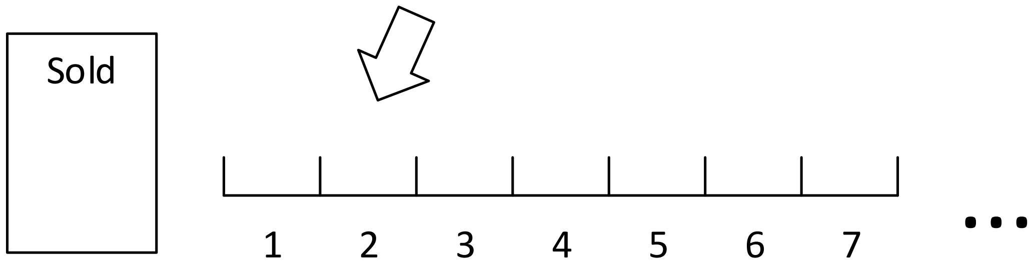
The last person puts an extra bag at location 3.



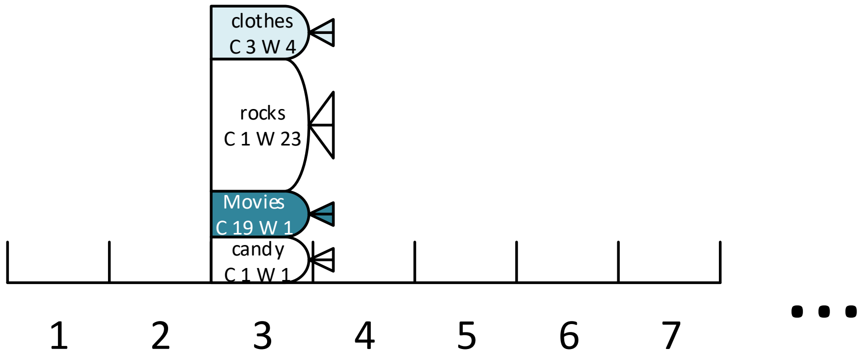
Case 2
Initially we have no bags



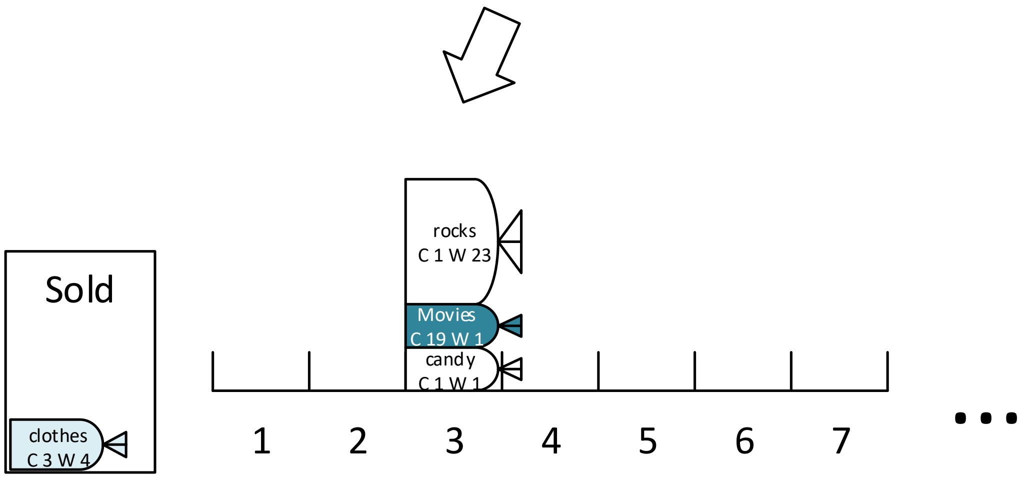
The first person tries to buy bags from an empty location (because all locations are empty).



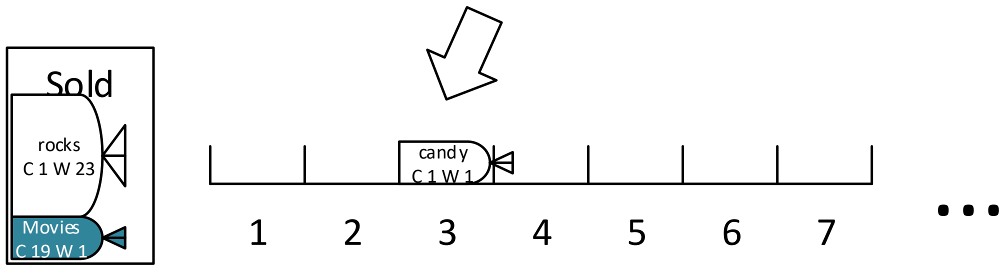
The next person drops off some bags at location 3.



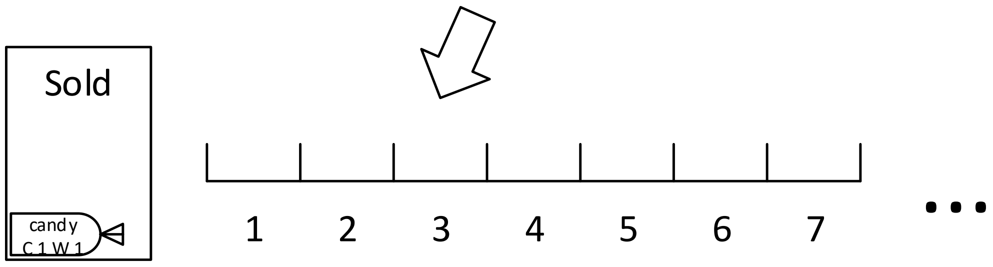
The next person begins to buy things from location 3.



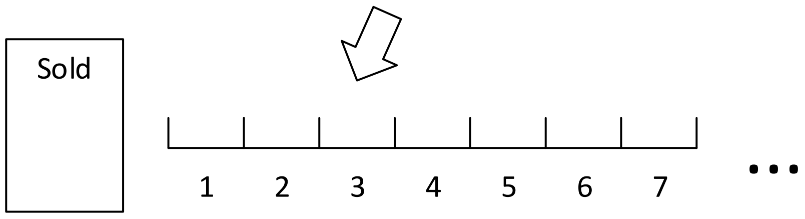
The next person continues to buy thing from location 3.



The fifth person buys the last of the items at location 3.



The last person tries to buy items from location 3 (to no avail).



Grading Information
Reading from standard input/output – 10 points

Create a linkable data structure that stores bag information – 10 points

Use a stack to store the drop off locations – 10 points

Use some method/data structure to store bags sold prior to printing to enable correct output order– 10 points

Good comments/variable names/whitespace usage – 10 points

Your program will be tested on 10 test cases – 5 points each

No points will be awarded to programs that do not compile.

Solutions without a linked data structure will receive a maximum of 50 points

Only cases that finish within the maximum of {5 times the judge solution, 10 seconds} will be graded.