Coding test - Products

The Brothers Inc. sells different products. We know the prices and selling quantities of each product on one day. Your tasks are the following:

1. Was there any unsold product?
2. Write a program that counts and lists those products which are more expensive than 8000 HUF.

Example (only the data, the explanations are not part of the input, communication with the user is not included):

Input	**Output**
6 *[count of products]*	Yes *[answer to Task 1]*
9000 1 *[price and quantity of the 1st product]*	2 1 3 *[answer to Task 2]*
8000 2 *[price and quantity of the 2nd product]*	
8500 0 *[price and quantity of the 3rd product]*	
7200 3 *[price and quantity of the 4th product]*	
6800 4 *[price and quantity of the 5th product]*	
5300 5 *[price and quantity of the 6th product]*	

Specification

Input: $pCount \in \mathbf{N}$
Product = Price x Quantity; $Price \in \mathbf{N}$, $Quantity \in \mathbf{N}$
$Products_{1..pCount} \in \mathbf{Product}^{\mathbf{pCount}}$

Output$_1$: $Exists0 \in \mathbf{L}$

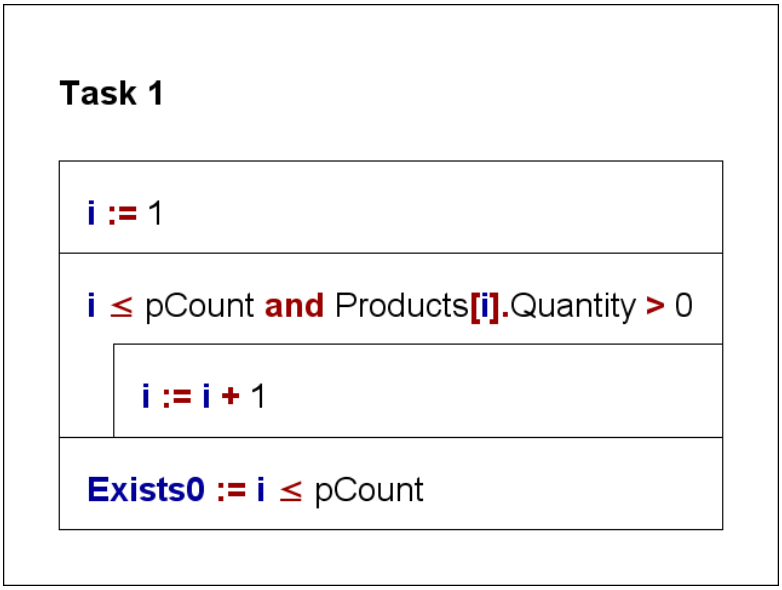
Output$_2$: $Count8000 \in \mathbf{N}$, $More8000_{1..pCount} \in \mathbf{N}^{\mathbf{pCount}}$ *(only using the first Count8000 elements)*

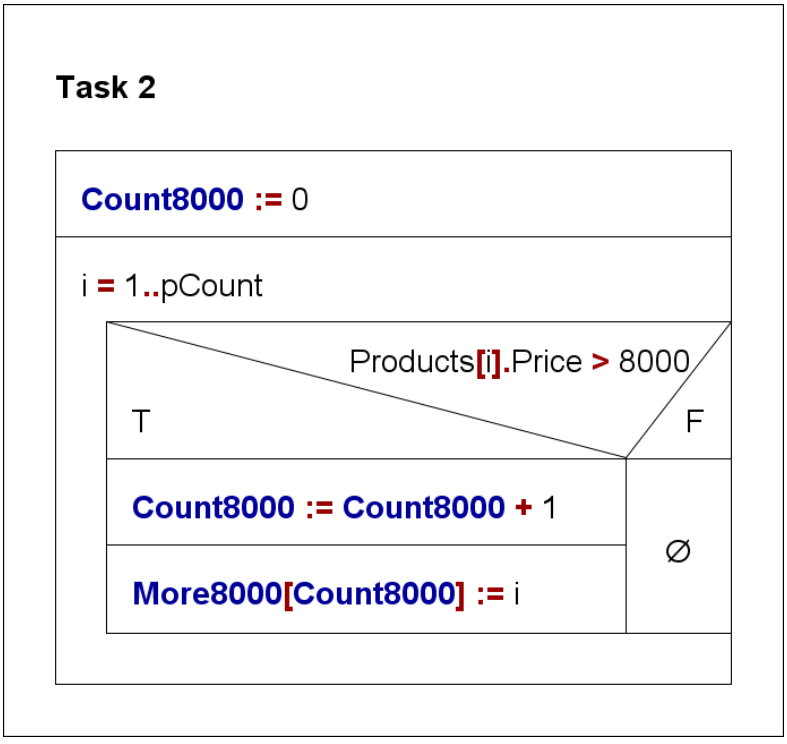
Precondition: $1 \leq pCount \leq 100$ and $\forall i \in [1..pCount]$: $Products_i.Price \in [0..100000]$ and $Products_i.Quantity \in [0..1000]$

Postcondition$_1$: $Exists0 = \exists i \in [1..pCount]: Products_i.Quantity = 0$

Postcondition$_2$: $Count8000 = \sum_{\substack{i=1 \\ Products_i.Price > 8000}}^{pCount} 1$ and $More8000 \subseteq \{1, \ldots, pCount\}$ and $\forall i \in [1..Count8000]: Products_{More8000_i}.Price > 8000$

Algorithm


Task 1
i := 1
i ≤ pCount and Products[i].Quantity > 0
i := i + 1
Exists0 := i ≤ pCount


Task 2
Count8000 := 0
i = 1..pCount
Products[i].Price > 8000
T
F
Count8000 := Count8000 + 1
More8000[Count8000] := i
∅

Evaluation and instructions

1. Your code should not contain syntax errors (it should be able to compile), otherwise you will get 0 points.
2. If there is a runtime error for any valid input, you will get half of the points.
3. The first two lines of your code should contain the date, the topic of your task and your name and Neptun code in a comment.
4. You need to write the input reading and output writing based on the specification. You have to include **appropriate communication with the user** (messages describing what you ask for and what you print)! *[Input: 5 points, Output task 1+2: 5+5 points]*
5. When reading the input, you should perform input error checking according to the precondition. *[10 points]*
6. You should follow the given algorithm and use the exact same variable names as in the algorithm and the specification. *[Task 1+2: 10+10 points]*
7. Use the data structures described in the specification to store the data. *[5 points]*
8. The two tasks are connected, you need to read the input only once and solve the two tasks with the same data.