Take home programming problem

You are to write a program to implement a cashier line simulation. This program should read input from a file and print the resulting score to the console. (The program should be a console-only program.) The program should take a single command line parameter, the name of the input file.

Customers arrive at a set of registers to check out. Each register is run by a cashier who serves customers on a first-come, first-served basis. The goal of the problem is to determine when all customers are done checking out.

One of the most important criteria for a successful solution is that it correctly solves the problem, but it is also an opportunity for you to demonstrate your object-oriented knowledge and testing skills. We will determine this by both code inspection and acceptance testing. We will also be considering your overall approach to the problem and your programming style. This assignment is an opportunity to show off your design, testing, and object-oriented skills.

Deliverables

1) Source code in one of the following languages:
    a. C# .NET and Visual Studio (freely available at http://www.visualstudio.com/en-us/products/visual-studio-community-vs.aspx)
    b. C++
    c. Java
    d. Python
2) Any other supporting code or resource(s) you used to develop and test your solution

Problem

1) The number of registers is specified by the problem inputs; registers are numbered 1, 2, 3...*n* for *n* registers.
2) Time is measured in minutes.
3) The grocery store always has a single cashier in training. This cashier is always assigned to the highest numbered register.
4) Regular registers take one minute to process each customer's item. The register staffed by the cashier in training takes two minutes for each item. So a customer with n items at a regular register takes *n* minutes to check out. However, if the customer ends up at the last register, it will take 2*n* minutes to check out.
5) The simulation starts at t=0. At that time all registers are empty (no customers in line).
6) Two types of customers arrive at the registers:
    a. Customer Type A always chooses the register with the shortest line (fewest number of customers in line).
    b. Customer Type B looks at the last customer in each line, and always chooses to be behind the customer with the fewest number of items left to check out, regardless of

how many other customers are in the line or how many items they have. Customer Type B will always choose an empty line before a line with any customers in it.

7) Customers just finishing checking out do not count as being in line (for either kind of Customer).
8) If two or more customers arrive at the same time, those with fewer items choose registers before those with more, and if they have the same number of items then type A's choose before type B's.

Input

Input is in the form of a single integer (number of registers), followed by a list of pairs. Each pair specifies the time in minutes (from a fixed offset) when a customer arrives to the set of registers, and how many items that customer has. Each pair appears white-space separated on a line by itself in the input file; see below for a sample input and output.

Example 1

For the following input file:

```
1
A 1 2
A 2 1
```

The following highlights occur:

- t=0 : Simulation starts with one register, which is a training register.
- t=1 : Customer #1 (type A) arrives with 2 items and goes to register #1 which starts servicing him.
- t=2 : Customer #2 (type A) arrives with 1 item and goes to register #1, behind
- Customer #1.
- t=3 : (Customer #1 now has one item left, since the first item took two minutes).
- t=5 : Customer #1 leaves and register #1 starts servicing Customer #2.
- t=7 : Customer #2 leaves.

Here is the expected command output:

```
C:\>grocery.exe input.txt
Finished at: t=7 minutes
```

Example 2

For the following input file:

```
2
A 1 5
B 2 1
A 3 5
B 5 3
A 8 2
```

The following highlights occur:

- t=0 : Simulation starts with two registers; #2 is a training register.
- t=1 : Customer #1 (type A) arrives with 5 items and goes to register #1 which starts servicing him.
- t=2 : Customer #2 (type B) arrives with 1 item and goes to register #2 which starts servicing her.
- t=3 : Customer #3 (type A) arrives with 5 items. Since he is type A, he goes to register #1 (lowest number of two with equal number of customers), behind Customer #1.
- t=4 : Customer #2 leaves from register #2 (which took two minutes). (At this point, register #1 has 7 items total: 2 for Customer #1 and all 5 for Customer #3.)
- t=5 : Customer #4 (type B) arrives with 3 items. Since register #1 has customers with 6 items, and register #2 is empty, she goes to register #2 which starts servicing her.
- t=6 : Customer #1 leaves and register #1 starts servicing Customer #3.
- t=8 : Customer #5 (type A) arrives with 2 items. Since both registers have one person, she goes to register #1 behind Customer #3.
- t=11 : Customer #3 leaves and register #1 starts servicing Customer #5.
- t=11 : Customer #4 leaves from register #2.
- t=13 : Customer #5 leaves from register #1.

Here is the expected command output:

```
C:\>grocery.exe input.txt
Finished at: t=13 minutes
```

Example #3

For the following input file:

```
2
A 1 2
A 1 2
A 2 1
A 3 2
```

Here is the expected command output:

```
C:\>grocery.exe input.txt
Finished at: t=6 minutes
```

(Note that this example illustrates the requirement that departing customers aren't counted in line).

Example 4

For the following input file:

```
2
A 1 2
A 1 3
A 2 1
A 2 1
```

Here is the expected command output:

```
C:\>grocery.exe input.txt
Finished at: t=9 minutes
```

(Note that this example illustrates the requirement that customers with fewer items choose lines sooner).

Example 5

For the following input file:

```
2
A 1 3
A 1 5
A 3 1
B 4 1
A 4 1
```

Here is the expected command output:

```
C:\>grocery.exe input.txt
Finished at: t=11 minutes
```

(Note that this example illustrates the requirement that customers of type A choose before customers of type B).