C++ OOP – Exam

Gas Pumps

You are the developer of a gas company that must operate throughout a gas supply crisis.

Due to political instability, the previous management illicitly destroyed a crucial file(s) in your gas control pump module. Fortunately, they left some of the files, which allows you to restore the management module to its full functionality.

Your task is to study the code and implement the missing functionalities.
You must also parse your input (read it from the standard input stream).

Your program is also given several commands to execute.
After you execute the command, you should print to the standard output the outcome of the command, which is the new capacity of the gas storage or an error in case the storage is already empty.
The possible commands are:

```
enum class Commands {
  INIT, // Init Q
  IN, // GasIn Q T
  OUT, // GasOut Q T
  DEMAND, // DEMAND Q T
  END // End
};
```

Where:

- ✓ `INIT` – sets the initial quantity of Q cubic tons in the storage.
- ✓ `IN` – Pumps in Q cubic tons for T hours, adding X*Y cubic tons to the storage. Indicates the amount of gas added.
- ✓ `OUT` – delivers Q cubic tons for T hours, reducing the quantity by X*Y tons
- ✓ `DEMAND` – Analyzes the demand of Q cubic tons for T hours, showing the check result as per the examples below.
- ✓ `END` – terminates the control sequence, ends the program

Input

- Commands, as specified, one command per line, with or without parameters.

Output

- Each command has its respective output format (see below). Some commands may have more complex output, depending on the current state.
- Each output starts with "Hour NNNN:", where NNNN is the number of hours passed after executing each command, which has the hours amount for execution (the Demand command is only a quantity check, it does not take time to execute)
- The rest of the line depends on the output of each command.
- The GasOut command output varies depending on whether it fully satisfies the requested quantity. If the requested quantity is not satisfied, it also prints out the shortage.

Restrictions

Time limit: 250ms (0.255s)
Memory limit: 16 MB

Example 1:

Input	Output
INIT 20000 GASIN 5 30 GASOUT 200 2 GASOUT 20000 1 GASOUT 1 1 GASIN 20 1 END	Hour 0000: Gas quantity: 20000 kg3 Hour 0030: Pumping in 5 kg3 for 30 hours, remaining 20150 kg3 Hour 0032: Delivering out 400 kg3, remaining 19750 kg3 Hour 0033: Delivering out 19750 kg3 (shortage 250 kg3), remaining 0 kg3 Hour 0034: Gas Storage Empty. Hour 0035: Pumping in 20 kg3 for 1 hours, remaining 20 kg3 Hour 0035: Final gas quantity: 20 kg3

Example 2:

Input	Output
INIT 0 GASIN 50 30 DEMAND 200 2 GASOUT 400 5 DEMAND 200 2 GasIn 20 60 Demand 40 3 Demand 600 30 GasOut 20000 1 GasOut 1 1 GasIn 2000 5 End	Hour 0000: Gas quantity: 0 kg3 Hour 0030: Pumping in 50 kg3 for 30 hours, remaining 1500 kg3 Hour 0030: CHECK: OK. Hour 0035: Delivering out 1500 kg3 (shortage 500 kg3), remaining 0 kg3 Hour 0035: CHECK: Gas Storage Empty. Hour 0095: Pumping in 20 kg3 for 60 hours, remaining 1200 kg3 Hour 0095: CHECK: OK. Hour 0095: CHECK: Shortage of 16800: availability 40 for 30 hours Hour 0096: Delivering out 1200 kg3 (shortage 18800 kg3), remaining 0 kg3 Hour 0097: Gas Storage Empty. Hour 0102: Pumping in 2000 kg3 for 5 hours, remaining 10000 kg3 Hour 0102: Final gas quantity: 10000 kg3