Bank Management

Problem Statement

You are the manager of a bank. At the beginning, before doing any transaction you have a certain amount of reserve money. Let's call this amount INITIAL_RESERVE. Then people come to your bank to make transactions. Each comprises of two properties:

1. `Timestamp` : An integer number indicating the time stamp of the transaction.
2. `Transaction Type` : It will be either Withdraw or Deposit
3. `Amount` : The amount of transaction.

When someone makes a deposit that amount of money is added to bank's reserve. When someone withdraws money that amount of money is paid by the bank from the reserve. All the transactions have to be processed in the order of timestamp. If the bank doesn't have enough reserve to make the withdrawal then the transaction is declined.

At the beginning you will be provided with the amount of INITIAL_RESERVE and the list of transactions. Then you will have to answer a few queries. There will be only one type of query. It is described below.

`Query start_time end_time` : As a bank manager you feel like it would help you if you could process tha deposits before withdrawals. That is exactly what you will be allowed to do here. You will be allowed to process all the deposit transactions before any withdrawal within the start_time and end_time range (start_time inclusive, end_time exclusive). But you still can not chnage the ordering of transactions within deposit (or withdrawal). Meaning, you can choose to process a deposit before a withdrawal within this range but all the deposit transactions must be processed in the order of timestamp. The same with withdrawal.

As a result, it might allow you to process some of the transactions that you previously declined. You have to output the maximum number of declined transactions you can now process as a result of this reordering transaction processing.

`Clarification` : The reordering of one query has no effect on another query. Which means at the beginning of each query you must start with the initial ordering of transactions and then reorder in the given range.

Task 1

Sample Input

First line of input contains three integers indicating the INITIAL_RESERVE and NUMBER_OF_TRANSACTIONS and NUMBER_OF_QUERIES. Then NUMBER_OF_TRANSACTIONS lines follow describing the transactions. Which is followed by NUMBER_OF_QUERIES lines follow describing the queries.

```
1000 11 3
5127 Deposit 500
8223 Deposit 1500
12309 Withdraw 2000
12310 Withdraw 3000
12311 Deposit 2000
15127 Deposit 5000
82044 Deposit 2500
123097 Withdraw 10000
224109 Withdraw 3000
333444 Deposit 30000
333445 Withdraw 50000
Query 100000 500000
Query 5000 12000
Query 12309 12312
```

Sample Output

```
2
0
1
```

Task 2

Sample Input

In the second task the queries and transactions can come in random. First line of input contains one integer indicating the INITIAL_RESERVE. Then each line may indicate a transaction or a query. Each query and transaction will have the aforementioned format.

```
1000 11 3
5127 Deposit 500
8223 Deposit 1500
12309 Withdraw 2000
12310 Withdraw 3000
Query 5000 12000
12311 Deposit 2000
Query 12309 12312
15127 Deposit 5000
82044 Deposit 2500
123097 Withdraw 10000
224109 Withdraw 3000
Query 100000 500000
333444 Deposit 30000
Query 100000 500000
333445 Withdraw 50000
```

Sample Output

```
0
1
0
2
```

Assumptions

1. `Timestamp` will come in an ascending, sorted way and they will all be different.

Expectations

1. Program needs to be fast enough to handle all functionalities.
2. Only use the standard libraries your programming language provides.

Additional Tasks

1. Generate the test input files using a program. Test your solution with the input files. Create at least 5 input files. At least one of the input files should contain a minimum of 100000 transactions and 100000 queries.

2. Dockerize your solution to make it deployable in any environment.

Submissions Guidelines

1. Create a github repository with all the codes (including solution to the problems and input generator).
2. Read the inputs from files `input_#.txt` and write the outputs to files `output_#.txt` . Where # is replaced by a number.
3. Write the documentation and running instructions in the `README.md` file.
4. Solve as many tasks as you can.
5. If you have difficulties understanding problem statement then use your best guess.
6. Try to write maintainabale code.