Assignment 3 - Logging and Recovery

Note: *Strictly follow input/output formats. Make sure your code outputs only what is asked, i.e. no extra outputs such as debug prints. Evaluation will be automated. Deadline is 9:00 pm, 10th April 2017.*

Part 1: Undo Logging

Write Undo logs into a file for a given set of transactions.

Input file format:
The first line of the file will be a list of database element names and their initial values, space separated, on a single line. For example:

```
A 4 B 4 D 5
```

Each transaction will begin on a new line, with the transaction name and number of actions in the first line, followed by actions of form read(), write(), output(), or an operation in successive lines. For example:

```
T1 4
read(A, t)
t := t+2
write(A, t)
output(A)
```

Instructions:
Assume the transactions are executed in a round-robin fashion with another command-line argument ‘x’. This means you carry out *x* actions in the first transaction, then *x* actions in the second, and so on.

Write UNDO logs for the set of transactions, given the file name and *x* as command-line arguments, into standard output. For example, if you’re doing the assignment in Python and name it *undo.py*,

```
python undo.py t1.txt 5
```

The above command should generate logs in the following format for transaction starts, database modifications, and transaction commits. After every log entry, print the list of database elements and their values, space-separated, in lexicographic order. For example:

```
<T1, start>
A 4 B 4 D 5
<T1, A, 8>
A 4 B 4 D 5
<T1, commit>
A 8 B 4 D 5
```

Part 2: Undo Recovery

Given an input file containing an UNDO log till a crash point, and the current set of database element values, perform a recovery - output the set of database elements and their recovered values.

Input file format:
First line is a space-separated list of database elements and their values, space separated. For example,

```
A 4 B 4 D 5
```

This is followed by a number of log statements which are either starts, database modifications, commits, non-quiescent checkpoint starts, checkpoint ends in successive lines. For formats, check the example:

```
<T1, start>
<T3, start>
<T1, A, 8>
<start ckpt T1, T3>
<T2, start>
<T1, commit>
<T3, D, 10>
<T3, commit>
<end ckpt>
```

And so on. The last log entry in the file is the entry just before the crash happened.

Instructions:
The input file is given as a command-line argument. If you're writing in Python, and the code file is *recover.py*,

```
python recover.py log.txt
```

The above command should output a single line containing the list of database elements and their values after recovery, space-separated, in lexicographic order. (Example: `A 4 B 4 D 5`)

Submission format:

Code from part 1 named as *rollnumber_1.ext* (Example: 201402160_1.py or 201402160_1.out), and part 2 as rollnumber_2.ext (201402160_2.py).

A bash script *rollnumber.sh*, which, if given two command-line arguments runs logging (The two arguments being input file and *x*), and if given only one argument, runs recovery, and produces the output.

Zip the three files - *rollnumber_1.ext, rollmumber_2.ext, rollnumber.sh* (with additional required files if any) into *rollnumber.zip*. Upload the zip.