CS 304 Assignment 1

You are required to implement a buddy memory manager and simulate its working. You will be given the upper and lower sizes admissible for blocks in the system and a list of requests.

A request is made for a process and it may either be for a block of a certain size or just an indication of termination. Requests should be attended on a first come first served basis. After serving all requests, your program should display the state of the buddy system at that point, indicating which processes are in memory and which blocks are free.

Notice that, whenever there is a request that corresponds to a block of size s, your program should select the block of that size that was most recently declared free. Furthermore, when a block is split in two, the left-one (lower addresses) should be selected before the right-one. You can assume that the list of requests is such that all requests can always be served. In other words, you can make the following assumptions: no process will request more than the available memory; processes are uniquely identified while active; and no request for process termination is issued before its corresponding request for memory allocation.

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

The first line of input consists of two numbers, U and L, that determine the upper ($2^U k$) and lower ($2^L k$) block sizes admissible. You can assume U > L > 0. The following input lines are requests being made, one per line. A request is defined by two values, P and S, where P is a capital letter that identifies the process associated with the request, and S >= 0 is a number. If S > 0 then it is a memory block request; Otherwise if S = 0 it is a request indicating that process P has terminated.

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.
The output must list the state of the buddy immediately after having served the last request. This corresponds to listing the processes still in memory, if any, and the free blocks available. Processes and free blocks must be listed in a left to right order as you traverse the buddy (i.e. from lower addresses towards upper memory addresses).
The output format for processes is P : S, P is the process identifier and S its size, and for free blocks is Free Block :S where S is the size of the free block.

You are required to show the working for 5 different values of U and L and 5 different input request patterns.

Implementation should be in C/C++. The submitted zip folder should contain the source code along with input files and a README file describing the usage.

Sample Input

```
1

10 4

A 70
B 35
C 80
A 0
D 60
B 0
```

Sample Output

```
Free Block: 128
Free Block: 64
D: 60
C: 80
Free Block: 128
Free Block: 512
```