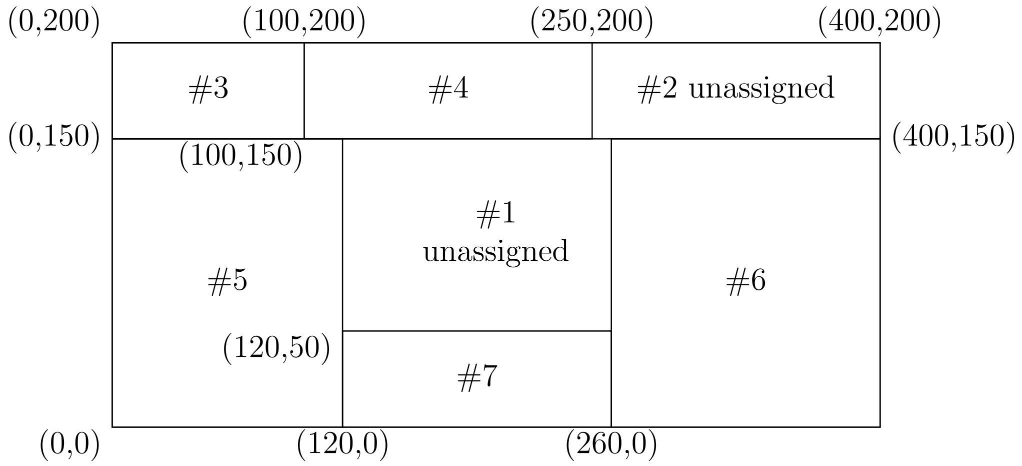
5. One of the functions performed by a typical GUI-building package (GUI: Graphical User Interface) is to lay out a set of subwindows within some larger window (in Java, for example, this is the job of `LayoutManagers`). In this problem you'll build part of such a facility. The basic units we deal with are what we'll call *windows,* each of which is a rectangle whose sides are either vertical or horizontal (no slanting lines). A window may be a *subwindow* of a another, in which case it lies entirely inside the other (which is called its *parent*).

Here, we'll do a simplified layout manager. We start with an empty window containing no children. At each point in the process, there will be a rectangular, unassigned area of the window, initially all of it. We can add a child window to the top, bottom, left, or right of this unassigned area. When adding a window to the top or bottom, we must specify its height, but not width. When adding to the left or right, we specify width, but not height. The idea is that the child is positioned at the indicated side of the unassigned area, and that it expands in width (on the top and bottom) or height (on the left or right) to fill the whole side of the unassigned area. After doing this, we end up with a hierarchy of nested windows. We'll assume there is one outermost window containing all the rest. To complete the process, we specify both a height and width for this outermost window.

For example, consider 7 windows: #1 is the outermost, size 400 × 200 (width times height). First, we place window #2 with height 50 on the top of #1. Then we put window #3 with width 100 on the left of #2. Then we place window #4 with width 150 on the left of the remaining space of #2. Next, we put window #5 with width 120 on the left of the remainder of window #1. Then, window #6 on the right of #1 with width 140. Finally, we put window #7 at the bottom of #1 with height 50. Here is the result:



Window #2 here encloses #3 and #4, and window #1 includes all the rest. Windows #3–7 are all unassigned space, since they contain no children.

Input to your program is in free form. It begins with two integers, W, and H, the width and height of window #1, which is the outer window. Next there are entries for each of the remaining windows, each of the form

$N \; P \; s \; d$

where N is the number of the window, P the number of its parent, s the dimension (width or height, depending on placement) of the window, and d is one of the letters `T`, `B`, `L`, or `R`, for top, bottom, left, or right. Assume that window numbers are consecutive starting at 2 for the first non-top-level window, and that parents always precede children.

The output consists of a list of the number, coordinates of the lower-left corner, width, and height of each window. Use the format shown in the example below, which shows the input and results from the preceding example.

Input	Output
`400 200`	`1. 400x200 @ (0,0)`
`2 1 50 T`	`2. 400x50 @ (0,150)`
`3 2 100 L`	`3. 100x50 @ (0,150)`
`4 2 150 L`	`4. 150x50 @ (100,150)`
`5 1 120 L`	`5. 120x150 @ (0,0)`
`6 1 140 R`	`6. 140x150 @ (260,0)`
`7 1 50 B`	`7. 140x50 @ (120,0)`