Assignment 3: Simulating the recursion stack for quick sort.

Call the instantaneous description of a stack at time t to be the contents of the stack at t, with the leftmost element being the top the stack and the rightmost being the bottom. (Recall the discussion in the class).

Implement the recursive randomized quick sort algorithm. Create a stack, using a singly linked list. Each record in the stack would contain three integers: the two array indices (lo, hi) and the line number to which the program has to return after a recursion call is completed.

Maintain the stack as follows: For each call of recursion, push a record with the corresponding invocation parameters (the array indices and the return address) into the stack and when the function returns, pop the record. You can get an idea about this here:

`http://pages.cs.wisc.edu/~vernon/cs367/notes/6.RECURSION.html`

Input: An unsorted list of numbers.

Output: The stack modification history in terms of instantaneous descriptions of the stack, one per line, with appropriate indentation–same levels of recursion are indented equally:

```
{(lo1,p-1)}
        {(lo1,hi2-1), (lo1,p-1)}
                  {....}
                     .
                     .
                     .
{(p+1,hi1)}
          {(....),(....)}
                  {....}
                     .
                     .
     .
```

...and of course the sorted array finally.

Caution: The number of instantaneous descriptions will blow up quickly as the number of elements in the input array increases!

Important: You have to maintain an explicit stack, implemented as a singly linked list.