CS213M: Assignment 2

Problem 3: A Simple Text Editor

Due Date: 04/02/2015

We implement a (very simple) text editor in this problem. Our editor always starts with an empty string. We refer to the current contents of the editor with `S`. It supports the following operations.

1. `append(W)`

   Appends the string `W` at the end of string `S`.

2. `erase(k)`

   Erase the last `k` characters of `S`. `k` will be a positive integer less than or equal to the length of `S`.

3. `get(k)`

   Prints out the k^{th} character of `S`. We number the characters from 0. `k` will be a non positive integer less than the length of `S`.

4. `undo()`

   Undo the last not previously undone operation (either `append()` or `erase()`) and revert back to the state before that operation.

5. `redo()`

   When `redo()` is called, it will redo an operation that satisfies the following
   1. it is either as `erase()` or an `append()` operation,
   2. it was previously undone,
   3. after it was undone, there have been no `append()` or `erase()` operations, and
   4. all the operations that were undone after this one have been redone already.

   If an operation satisfying the above criteria does not exist, `redo()` will not do anything

Input Format

The first line contains `Q`, the number of operations.
Each of the following `Q` lines starts with a character `t`, denoting the type of operation listed in the problem statement. Then, `t` is followed by the argument of the operation, if it has any, separated by a space.
The character used to denote an operation will the lower case first character of the name of the operation. For example, `a` will used to denote `append()` and `r` will be used to denote `redo()`

Output Format

For each `get()` operation, print a single line with the returned character of that operation.

Examples
Example 1

Command	String S after the command is executed
a abcdefg	abcdefg
e 6	abcdef
a pq	abcdefpq
a r	abcdefpqr
g 6	abcdefpqr
u	abcdefpq
g 3	abcdefpq
u	abcdef
r	abcdefpq
g 1	abcdefpq
u	abcdef
a s	abcdefs
r	abcdefs
u	abcdef
g 4	abcdef

The output of the above example will be

```
p
d
b
e
```

PS: If you are wondering how this is a very very primitive and inconvenient editor we just implemented, do have a look at the (very old) editor **ed**. It can be found on any standard Linux installation.