**General Note:**

- Read and understand the question carefully.
- For implementation, you are permitted to use the code that you have submitted as an assignment.
- You are not permitted to use global variables and/or static variables.
- Assume that all the inputs in the test cases are valid.

1. SBB bank is starting a new locker facility with $n$ lockers, numbered from $1$ to $n$, for its customers. You are asked to create a locker management system for the bank staff to store the details of the customers who use the locker facility. The system should allow the staff to

   - Assign a customer to a locker. Staff can also replace a customer already assigned to a locker with a new customer.
   - Free a locker by removing the customer from the locker.
   - If the staff makes a mistake while doing the above operations (assign/free), he/she should be able to undo that operation. At a time, an undo can revert only the last performed operation.

   The locker management system uses an array $A$ of size $n$ to store the account number of customers who are provided with locker facility. $A[i]$ stores the account number of the customer assigned the locker number $i+1$. Initially, all the lockers are empty and assigned with an account number $-1$. Use a stack, implemented using a **linked list**, to keep track of the assign and free operations performed.

   Your program should implement the following functions as per the given function prototypes:

   - $main()$: Repeatedly read a character '$i$', '$d$', '$p$', or '$u$' from the console and perform the corresponding operations given in the section Input/Output Format, using the following functions, until character '$t$' is encountered.
   - $assign\_locker(A, S)$: Read the account number $c$ and locker number $p$ of a customer, and insert $c$ in the array $A$ based on $p$. Also store the details of $insert$ operation in the stack $S$. [1.5 marks]
   - $free\_locker(A, S)$: Read the locker number $p$ to be freed. In array $A$, remove the customer from locker number $p$. Also store the details of $delete$ operation in the stack $S$. [1.5 marks]
   - $undo(A, S)$: From the stack $S$, get the details of the last performed operation in the array $A$, and undo that specific operation. [2 marks]
   - $print\_details(A, n)$: Print the account number of all the customers assigned with a locker in the array $A$ of size $n$ in the increasing order of locker number, separated by a space. If there are no lockers assigned to any customers at present, print -1. [1 Mark]

### Input/Output Format

The input consists of multiple lines. First line of the input contains an integer $n \in [1, 10^5]$ representing the total number of lockers.

Each subsequent line starts with a character from $\{i, d, p, u, t\}$ followed by at most two integers.

- Character '$i$' : Character '$i$' will be followed by two integers, $c \in [1, 10^6]$ and $p \in [1, n]$ representing the account number and locker number of a customer, respectively. Assign the customer to the locker number $p$ using $assign\_locker()$ function.
- Character '$d$' : Character '$d$' will be followed by an integer $p \in [1, n]$ representing the locker number. Remove the customer assigned with the locker number $p$ using $free\_locker()$ function.
- Character '$p$' : Print the account number of all the customers using $print\_details()$ function.
- Character '$u$' : Undo the last performed operation using $undo()$ function.
- Character '$t$' : Terminate the program.

**Sample Input and Output**

**Input 1**

```
6
p
i 110110 5
i 210036 3
i 230412 1
p
u
p
i 177455 5
i 455341 1
p
d 5
p
t
```

**Output 1**

```
-1
230412 210036 110110
210036 110110
455341 210036 177455
455341 210036
```

**Input 2**

```
10
i 333452 8
i 212213 6
i 676212 6
p
u
```

```
p
u
p
u
p
i 676161 1
i 899899 3
d 3
d 2
i 555666 10
p
i 123111 2
i 124212 5
i 125516 7
i 777126 7
p
u
p
t
```

**Output 2**

```
676212 333452
212213 333452
333452
-1
676161 555666
676161 123111 124212 777126 555666
676161 123111 124212 125516 555666
```