Question 2

Write a program to implement a Hash Table data structure as described below.

You are given two hash functions. **f(x) = x mod m** and **g(x) = [x / m] mod m**

You should maintain two hash tables each of size ***m***. Given an element ***x***, **f(x)** is used to find ***index1*** which is the index of ***x*** in the first table, and **g(x)** is used to find ***index2*** which is the index of ***x*** in the second table. Each non-empty location in a hash table stores an element ***x*** and its alternate index in the other table.

Your program should include the following functions: (You can decide the function parameters based on your design.)

INSERT(x):

- Insertion of a new element is always done in the first table (at position ***index1***).
- If a collision happens (ie, when the index where an element to be stored is already occupied), the element is stored at that position after moving the element that is already present there to its alternate index in the other table.
- Resolve collisions until we find an empty space in either of the tables or ***m*** successive collisions happen. If an empty space could not be found even after ***m*** successive collisions, stop this process and discard the last element to be moved.

PRINT(T): Starting from index 0, print the data values stored in the hash table **T**, separated by a space. If an index is empty, print NIL

Please refer to the example given below.

Input data: 29 24 30 50 89 43 23
Hash table size: 5

The table given below contains the input elements and their indices, ***index1*** and ***index2*** calculated using the given hash functions **f(x)** and **g(x)**.

x	29	24	30	50	89	43	23
index1	4	4	0	0	4	3	3
index2	1	0	1	0	3	4	0

The columns of Table1 correspond to the entries in the Hash table1 after the insertion of each input element. Each non-empty location stores an element ***x*** and its alternate index in the Hash table2 in the format: *x, index2*.

Table 1		Element inserted, x						
		29	**24**	**30**	**50**	**89**	**43**	**23**
Index	**0**			30, 1	50, 0	30, 1	30, 1	30, 1
	1							
	2							
	3						43, 4	23, 0
	4	29, 1	24, 0	24, 0	29, 1	24, 0	24, 0	24, 0

The columns of Table2 correspond to the entries in the Hash table2 after the insertion of each input element. Each non-empty location stores an element ***x*** and its alternate index in the Hash table1 in the format: *x, index1*.

Table 2		Element inserted, x						
		29	**24**	**30**	**50**	**89**	**43**	**23**
Index	**0**				24, 4	50, 0	50, 0	50, 0
	1		29, 4	29, 4	30, 0	29, 4	29, 4	29, 4
	2							
	3							
	4							43, 3

Input / Output format:

- The first line contains an integer m corresponding to the size of the hash table.
- The next lines may contain a character from {i, p, t} followed by at most one integer.
- Character i followed by an integer separated by space: Insert the integer into the hash table using the function INSERT().
- Character p followed by an integer n ∈ {1, 2} (separated by a space):
    - If n = 1, print the values in the first hash table using PRINT().
    - If n = 2, print the values in the second hash table using PRINT().
- Character t: Terminate the program.
- The output of each command should be printed in a separate line.

Sample Input:

```
5
i 29
i 24
i 30
i 50
i 89
i 43
i 23
p 1
p 2
t
```

Output:

```
30 NIL NIL 23 24
50 29 NIL NIL 43
```