LRU

In this assignment, you will have to implement the replacement algorithm LRU(Least Recently Used). In this algorithm the block which is **unused** for the **longest** time gets replaced. Here, you will be given **n**(set associativity of cache), total number of indices in cache and a sequence of RAM **blocks**. You will have to find the sequence of HIT or MISS. Also, you will have to display the cache after each block access.

**Consider the cache as a 2d array where each row of the 2d array is a set and each column is an index of the cache. For example, if it is a four way set associative cache and it has 8 indices then this cache has 2 sets and each set is comprised of 4 indices. So, the dimension of the 2d array will be 2×4**.

**Input**

The **first line** contains two integers **n** and **m**. Here, **n** denotes the set associativity of the cache and **m** denotes the total number of indices in the cache. For example, n=4 and m=256 means it is a four-way set associative cache and it has 256 indices.

The **second line** contains a sequence of integers which denote the RAM blocks. The last integer of this sequence will be -1 to indicate the ending of this sequence.

**Output**

After each block access print “**HIT**” if it is a HIT otherwise “**MISS**”. Also print the cache after each block access. If a cell of the cache is empty then print “**B**”.

**Example**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 2 4  2 4 6 112 4 6 2 10 37 13 30 67 11 5 30 -1 | MISS   |  |  |  |  | | --- | --- | --- | --- | | 2 | B | B | B | | B | B | B | B |   MISS   |  |  |  |  | | --- | --- | --- | --- | | 2 | 4 | B | B | | B | B | B | B |   MISS   |  |  |  |  | | --- | --- | --- | --- | | 2 | 4 | 6 | B | | B | B | B | B |   MISS   |  |  |  |  | | --- | --- | --- | --- | | 2 | 4 | 6 | 112 | | B | B | B | B |   HIT   |  |  |  |  | | --- | --- | --- | --- | | 2 | 4 | 6 | 112 | | B | B | B | B |   HIT   |  |  |  |  | | --- | --- | --- | --- | | 2 | 4 | 6 | 112 | | B | B | B | B |   HIT   |  |  |  |  | | --- | --- | --- | --- | | 2 | 4 | 6 | 112 | | B | B | B | B |   MISS   |  |  |  |  | | --- | --- | --- | --- | | 2 | 4 | 6 | 10 | | B | B | B | B |   MISS   |  |  |  |  | | --- | --- | --- | --- | | 2 | 4 | 6 | 10 | | 37 | B | B | B |     MISS   |  |  |  |  | | --- | --- | --- | --- | | 2 | 4 | 6 | 10 | | 37 | 13 | B | B |     MISS   |  |  |  |  | | --- | --- | --- | --- | | 2 | 30 | 6 | 10 | | 37 | 13 | B | B |     MISS   |  |  |  |  | | --- | --- | --- | --- | | 2 | 30 | 6 | 10 | | 37 | 13 | 67 | B |     MISS   |  |  |  |  | | --- | --- | --- | --- | | 2 | 30 | 6 | 10 | | 37 | 13 | 67 | 11 |     MISS   |  |  |  |  | | --- | --- | --- | --- | | 2 | 30 | 6 | 10 | | 5 | 13 | 67 | 11 |     HIT   |  |  |  |  | | --- | --- | --- | --- | | 2 | 30 | 6 | 10 | | 5 | 13 | 67 | 11 | |

**\*\* You can use any programming language to implement this.**

**\*\* During evaluation I will check your code by giving different input. So, write your code in such a way that it can handle any input.**