Shah’s Memory

OK. Shah has a really weird memory. First of all, shah can only remember a certain number of elements (we will say n). When memorizing something new, Shah’s memory will have behave in 2 ways. First, if Shah’s memory is not full, it will simply remember it. If however, Shah’s Memory is full (aka has already n elements remembered), it will forget the element that was used last.

Shah can also try to remember the data that is in his memory. Shah’s memory allows shah to remember in two ways. First, if Shah simply tries to remember without looking for any specific element, it will return the element that was remembered most recently (and as you can guess, it will set this element as the most recently used). Secondly, Shah can try to remember a specific item that he may or may not have learned or may or may not have forgotten. When Shah tries to remember a specific item, it will either return the data or not (upto you how you wanna handle data not found). If Shah remembers the item successfully(aka it was in his memory) , then that item is the most most recently used.

Now when Shah is studying for an exam, he can manually tell his memory to forget stuff that he may not need by asking his memory to forget something and thus making room for more stuff to memorize. Forget also works in 2 ways. First, if Shah simply tries to forget something, it will forget the item that was used last/least recent. Secondly, Shah can forget using a specific item.

Shah’s memory is also generous to Shah and will try to help him out on his time of need when he's stressed (especially on exams!). When stressed, Shah’s memory allows Shah to not only remember data that is currently in memory, but also the last 10 items that were removed from his memory(either removed on purpose or evicted to make space for new items to be stored)

Your task, is to implement Shah’s Memory and efficiently ! We don't want shah waiting on his memory for too long! Implement it as a Class!!! Make use of all awesome and cool data structures you learned :)

NOTE: If an element is currently the most recently used and Shah calls another function that ends up making another element most recently used, then the first element (that was initially most recently used) , will become the second most recently used. So you will have to keep track of which data were used and somehow keep an order of when the elements were used. (Example below)

Here are some methods to help you get started:

```
/*
  -- This method will memorize the data
  -- And set it as the most recently used
  --  If the data already exists in memory, we will.
  --  simply make it was the most recently used.
```

```
*/
template <class Element>
void memorize (Element data);

/* Forget the element that was least recently used */
void forget ();

/* Forget the element passed in */
template <class Element>
void forget(Element data);

/* return the most recently used item */
template <class Element>
Element remember ();

/* return the element specified is it exists, otherwise return something that indicates this element
doesn't exist  ( your choice ) */
template  <class Element>
Element remember  (Element data);

/* see details above */
template<class Element>
Element stress (Element data);
```

Ex:
Assume we initialize Shah's memory is 4 elements:

```
memorize( 3)
        → [3,,,]
        → holds just 3, with 3 empty spots
        → most recently used is 3
        → from now on, just array will be shown with
        → most recent elements at lower indexes
memorize(1) → [1,3,,]
remember() → returns 1
remember(3)
      → return 3 and array is reordered: [3,1,,]
memorize (2) → [2,3,1,]
memorize (5) → [5,2,3,1]
memorize (1) → [1,5,2,3]  (1 becomes most recent)
memorize (9) → [9,1,5,2] ( 3 is evicted )
remember(3) → -1 ( I'm making ints positive only)
```

```
stress (3) →  3 and array is reordered: [3,9,1,5]
            -- we have the 10 most recently evicted
            -- items and 3 was in there.
            -- 3 becomes most recent used and
            -- 2 is evicted because it was least recenty
            -- used
forget  (3) → [9,1,5,]
            -- 3 is removed and the array is updated to
           --  the new most recently used elements.
           -- note we have one free spot now.
```