Stacks and Queues

The same limitations apply to the use of python lists (here called arrays) as in previous array assignments. It is OK to first get the implementation working, and then add the limitations.

- **Implement a stack using an array**
  - Make the class Stack
  - Implement the functions push(value) and pop() in the class
    - pop() should return a value
    - Both operations should be O(1) (amortized)
    - *Implement it so that there is no limit on the size of the stack*
      - *The program will have to re-allocate when the array fills up*

- **Implement a queue using an array**
  - Make the class Queue
  - Implement the functions add(value) and remove() in the class
    - remove() should return a value
    - Both operations should be O(1) (amortized)
      - First implement remove so that it is O(n)
      - Then use the circular buffer concept to make it O(1)
    - *Implement it so that there is no limit on the size of the queue*
      - *The program will have to re-allocate when the array fills up*
        - *Make sure this is not done unnecessarily often*

- **Implement a deque (double-ended queue) using an array**
  - Make the class Deque
  - Implement the functions push_front(value), push_back(value), pop_front() and pop_back() in the class
    - Pop_front and pop_back() should each return a value
    - All operations should be O(1) (amortized)
      - *Use the queue implementation as base for the circular buffer*
    - Implement it so that there is no size limit