void rotateToHead(int). Given a non-negative int argument (call it val), this function “rotates” the list to bring the first occurrence of val to the head of the list. Remember that rotation means that all elements “move” their position, with the end “wrapping around” to the head. If val is not present in the list, nothing happens.

void rotate(int). Given a non-negative int argument, this function “rotates” the list by this value. Rotation by x means that all elements “move” their position by x, with the end “wrapping around” to the head.
For example, suppose our list is 10 20 30 40. The function rotate(1) would lead to 40 10 20 30, while rotate(2) would lead to 30 40 10 20. Note that we can rotate by more than the length of the list, so rotate(6) would lead to 30 40 10 20. (You do not need to worry about negative inputs.)

void reverseTail(int): Given a non-negative int argument (say, val), this function reverses the order of the last val elements of the list, and leaves the others unchanged.
Suppose our list is 10 20 30 40. The function reverse(2) would lead to 10 20 40 30, while the function reverse(3) leads to 10 40 30 20. If the argument is more than the length of the list, we reverse the entire list.

bool palindrome(). The function returns true if the list is a palindrome, and returns false otherwise. A palindrome is a list where the first and last elements are the same, the second and second last elements are the same, etc.
Thus, the list 10 20 10 is a palindrome, while 10 10 20 is not. By default, the empty list and the list with one element are always palindromes.

Node∗ deletelast(int). Given an int argument, this function deletes the last occurrence of the element, and returns a pointer to the deleted node. It does not affect the order of anything else.
Thus, on deletelast(42) in the list 7 42 101 42 50 3, we get 7 42 101 50 3. (The usual delete would result in 7 101 42 50 3.)

void dedup(). This is a deduplication function, meaning that all duplicate values are deleted from the list. Specifically, for every value, you need to delete all subsequent occurrences from the list.
For example, suppose the list was (in order) 10 20 10 50 20. On deduplication, the list becomes 10 20 50. It is important that you do not shuffle the values. This function just performs this operation, so there is no argument.