Adding and Removing List Elements

Adding/Removing by Value

1. Create an empty list, then have the user enter five integer values and store them in the list. Find the sum and the product of all values.
2. Use a loop to create a list containing 100 random letters from A, B or C. Ask the user to enter one of the three letters, then remove all instances of that letter.
3. Given a list containing n positive integers and a positive integer k, remove all values that are multiples of k. That is, if $k=5$, then the list 3, 10, 7, 6, 20 becomes 3, 7, 6.
4. Read two values from the user, n and k, where $k < n$. Create a list containing n random integers. Determine the kth largest value in the list. For example, if the list contains 3, 5, -1, 2, 7, 12 ($n=6$), and $k=3$, then the 3rd largest value is 5. How can you do this without sorting the list?
5. *Statistics*: Given n positive integer values, create a vertical column chart using each value. For example, if the user enters 3, 2 and 4, your program should produce the following output.

```
      *
*     *
* * *
* * *
```

Adding/Removing by Index

6. *Card Selector*: Generate a list of five different playing cards (you may choose how to represent them). Display the user's hand in a nicely-formatted manner. Ask them which card (1-5) they would like to play, then remove that card from their hand. Display the new hand. Remember that lists in Python are zero-indexed!
7. Given a list of n integers, insert a zero entry after each even value. That is, the list 7, 2, 3, 4, 6, 5 would become 7, 2, 0, 3, 4, 0, 6, 0, 5.
8. In computer science, a *stack* is a data structure that stores elements according to two operations:
   - A new element may only be added to the top of the stack (last index).
   - An existing element in the stack may only be removed from the top (last index).

   Stacks are sometimes referred to as *Last-In-First-Out* (LIFO). A list can represent a stack by using the `append` and `pop` methods. Write a program that has the user add or remove elements from a stack. Display the contents of the stack after each user action. Display a message when the stack is empty (can no longer remove an element), or when it contains 10 elements (stack is full, cannot add any new elements).
9. Repeat the previous question, this time using a *queue*. A queue is a data structure such that:
   - A new element may only be added to the rear of the queue (last index).
   - An existing element in the queue may only be removed from the front (first index).

   Queues are sometimes referred to as *First-In-First-Out* (FIFO). As before, display the contents of the queue after each user action. Display a message when the queue is empty, or when it is full.
10. *Commencement Seating*: Create a program that will help organize staff seated on stage during this year's Commencement ceremony. In the first stage of your program, use a loop to input names of staff into a list, which represents a line of seats. In the second stage, use a loop to allow the user to reorder the staff until s/he is satisfied with the arrangement. To keep things user-friendly, your program should ask the user to enter the name of the staff to move, then to use one of three commands to determine their placement: "between X Y", "right of X", or "left of X", where X and Y are existing names in the list. You will need to use to string methods to help parse user input.