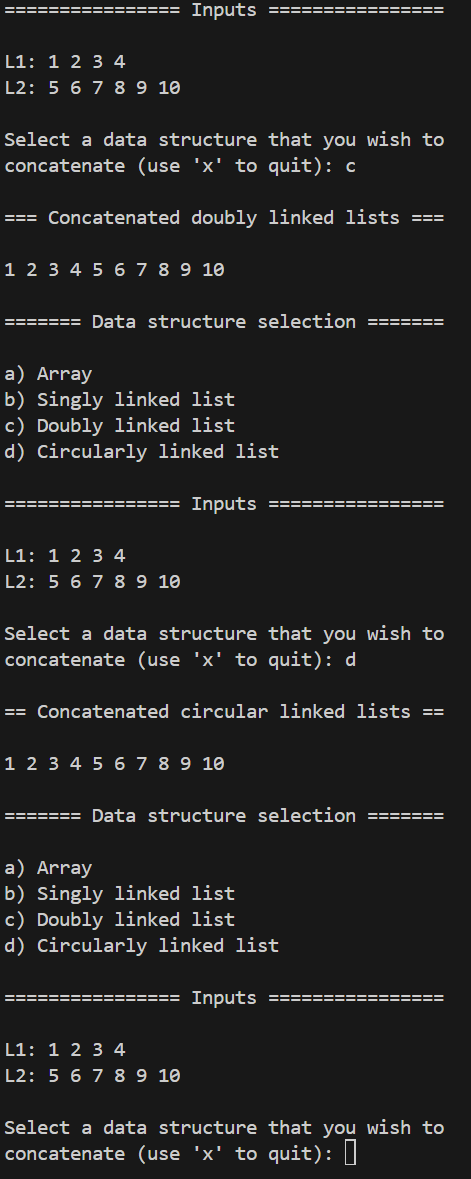
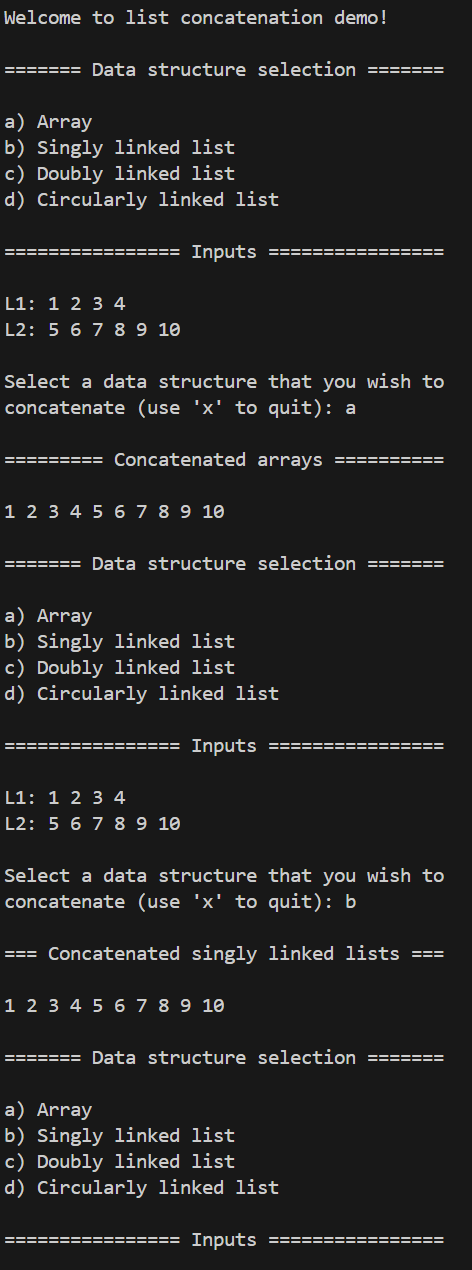
Submit screenshots of the results, code files, and a writeup describing your solutions in a zip file.



**Figure 1.** Console output for two lists of various types being concatenated.

**Array Concatenation:**

The array concatenation algorithm accepts an additional input array which will contain the concatenated output. This array is verified to be of an appropriate size given that the user passes necessarily passes the appropriate size as an input (the sum of the sizes of the two arrays to be concatenated). All elements from L1 are first copied into this buffer array. Following this, all elements of L2 are copied into the buffer array, beginning at the index indicating the size of L1. The algorithm has linear time complexity and constant space complexity.

**Singly Linked List Concatenation:**

The singly linked list concatenation algorithm uses L1 as the container for the concatenated list. Thus, L1 will be inherently modified during execution. Beginning at the front of L2, each node of L2 is appended to the end of L1. In each iteration, L2’s front is moved to its back. Upon completion, L2 will be identical to the original since the number of movements which occur is guaranteed to match the size of L2. The algorithm has linear time complexity and constant space complexity.

**Doubly Linked List Concatenation:**

The doubly linked list concatenation algorithm uses L1 as the container for the concatenated list. Thus, L1 will be inherently modified during execution. Beginning at the front of L2, each node of L2 is appended to the end of L1. In each iteration, L2’s front is moved to its back. Upon completion, L2 will be identical to the original since the number of movements which occur is guaranteed to match the size of L2. The algorithm has linear time complexity and constant space complexity.

**Circular (Singly) Linked List Concatenation:**

The circular linked list concatenation algorithm uses L1 as the container for the concatenated list. Thus, L1 will be inherently modified during execution. Beginning at the front of L2, each node of L2 is appended to the end of L1. In each iteration, L2’s cursor is advanced. Upon completion, L2 will be identical to the original since the number of movements which occur is guaranteed to match the size of L2. The algorithm has linear time complexity and constant space complexity.

**Future Improvements:**

The code can be improved through further polishing, more explicit commenting, as well as allowing the user to select their own input data.