1.

(a)

- Linked List is easy to insert / remove the node because the time complecity is O(1), while the array list’s time complecity to insert/remove the node is O(N)

(b)

- Accessing an item by its index, including get and set, can occur in O(1) time, but the Linked List take O(N) time because Linked List need to access entire list in order to access an item by its index.

2.

(a)

- O(N^2). This is because the adding to the front for ArrayList requires moving the other elements down, which takes O(N) time, and there are N iterations

(b)

- O(N). This is because the adding to the front for LinkedList requires O(1) time, with N iterations.

3.

(a)

- O(N^2). This is because the removing element for ArrayList must shift the items over, which is O(N), and the while loop makes N iterations

(b)

- O(N), This is because removing for LinkedList requires O(1) time with while loop N interations.

4.

(a)

- O(N^2). This is because Ist1 in the while loop take O(N), and inside the while loop, there is nested while loop, which takes O(N )for Ist2.

(b)

- O(N^2). This is because Ist1 in the while loop take O(N), and inside the while loop, there is nested while loop, which takes O(N) for Ist2.

5.

(a)

- O(N). This is because arrayList has time Complexity O(1) for getting the element, and for loop makes N interations

(b)

- O(N^2). This is because LinkedList has time Complexity O(n) for getting the element, and for loop makes N interations

6,

(a)

- The time complexity of removing each item from the front is O(N) with N interations. and time complexity of pop and push , which are stack operation, is O(1). In addition, inserting each item to the end of the arraylist is O(N) with N interations. Therefore, the worst case Big-O running time is O(N^2)

(b)

- The time complexity of removing, inserting of the items is O(1) with N interations. and time complexity of pop and push, which are stack operation, is O(1). In addition, inserting each item to the end of the Linkedlist is O(1) with N interations.Therefore, the worst case Big-O running time is O(N)

7.

input: a

stack:

output: a

input: +

stack: +

output: a

input: b

stack: +

output: a b

input: \*

stack: + \*

output: a b

input: c

stack: + \*

output: a b c

input: +

stack: +

output: a b c \*

stack:

output: a b c \*+

stack: +

output: a b c \*+

input: (

stack: + (

output: a b c \*+

input: d

stack: + (

output: a b c \*+ d

input: -

stack: + ( -

output: a b c \*+ d

input: e

stack: + ( -

output: a b c \*+ d e

input : )

stack: + (

output: a b c \*+ d e -

stack: +

output: a b c \*+ d e -

stack:

output: a b c \*+ d e - +