**ASSIGNMENT 2 REPORT**

By: Khadija Swailem 202301222

***Assumptions:***

I assumed stack is of unlimited size

***Design approach:***

**Attempt 1:**

Since the assignment stated that we use two stacks to implement a queue I drew the two stacks and drew the queue and used one stack for all the enqueue operations of the queue using push method, and then used the other stack for when I needed to dequeue so I popped all the elements from stack 1 and then pushed them in stack 2 , so now the order is reversed which will help implement the queue concept FIFO (first in first out) and then I popped the top of the stack as it uses LIFO (last in first out) and then popped all the elements from stack 2 and pushed them in stack 1 to reverse their order again back to original before dequeuing but without item we dequeued. I think this is the most optimum approach within the limits set as we move the elements once to stack 2 then pop the top to dequeue. And as for the enqueue operation I used push() function to enqueue it into the queue.

**Attempt 2:**

In the second attempt, since I had more freedom, I decided to use removeat(index) function from the linked list class to access the item at index 0 and insert(item,index).Based on the idea that we will use reverse only as a boolean flag and based on the state of the flag decide whether the stack is reversed or not and accordingly we can remove at index 0 for dequeuing and use append or insert for enqueuing, I created a function called setReverse() that changes the value of the Boolean flagrev variable so when its called its as if we reversed the stack to allow us to dequeue the item at index 0. To implement these changes and as advised I created a new class called Stack 1 that inherits from Stack class.Inside stack 1 I implemented a new push() function that checks the flag if its not reversed(flag=0) it will use append to add to the stack but if the stack is reversed(flag=1) it will use insert(0) at index zero to insert at the head.I also implemented a new pop() function that checks if the stack is reversed(flag=1) we can pop the item at index 0 using removeat(0), and if the stack is not reversed (flag=0) then call setReverse () on the stack object for example s1.setReverse() to “reverse” it and then pop using removeat(0) . I think this is an optimum approach because since I can directly access the element to be popped the time complexity is constant since also the item to be popped is at index 0 so it wont iterate over the elements.

***Big-O analysis:***

**Attempt 1:**

Time complexity of Enqueue operation: O(1). Since we just use push method which uses append and both have time complexity O(1).

Time complexity of Dequeue operation: O(n).Since we pop n times to get the elements out of stack 1 and then push n times to put them in stack 2 then we pop once to remove the top element(element to be dequeued) and then we pop n-1 times and push n-1 to return elements to stack 1 again to get their original order but after dequeuing.

**Attempt 2:**

Time complexity of Enqueue operation: O(1). Since we just use push method which uses append or insert at index zero they all have time complexity O(1), because even the insert at index zero won’t iterate over the nodes because its at index 0 and since we don’t have to shift the elements n times like in array implememntation whos complexity is O(n),its done in optimum time.

Time complexity of Dequeue operation: O(1).Since we call the pop function for one item and it uses removeat index zero which also has complexity O(1) and since we’re getting the item at index 0 it iterates only once so best case of iterating as we don’t iterate n times to reach index n so also O(1).