I certify that every answer in this assignment is the result of my own work; that I have neither copied off the Internet nor from any one else's work; and I have not shared my answers or attempts at answers with anyone else.

---

**1: Show how to implement a FIFO queue by means of two ordinary LIFO stacks so that the amortized cost of each ENQUEUE and DEQUEUE operation is O(1)(i.e., a sequence of n operations cost O(n)). Assume the actual costs for PUSH and POP are 1 and all other operations are free.**
**(a) First, write pseudo-code for the two operations using our notation and with adequate comments.**
**(b) Next, demonstrate the amortized costs using the accounting method.**
**(c) Now, demonstrate the amortized costs using the potential method.**

---

(a) Let the two LIFO stacks be referred to as stack1 and stack2

```
ENQUEUE(x)
   stack1.PUSH(x) | > Push x onto the top of the 1st LIFO Stack

DEQUEUE(x)
   if stack2 != EMPTY do
     stack2.POP(x)
     return x
      | > if the second LIFO stack isnt empty, than just pop off the top of the stack (x)
  else do
      for i = stack1.size down to 0 do
        stack1.POP(i)
        stack2.PUSH(i)
     stack2.POP(x)
     return x
     | > if the 2nd LIFO stack is empty, than pop off every element from the 1st stack
     | > Then push every element onto stack 2 (so that the order is reversed)
     | >Then pop off the last element in stack 2 which should have been the head of stack 1 (x)
```

(b)

| Opn | Actual Cost | Amortized Cost |
|---|---|---|
| POP(x) | 1 | 1 |
| PUSH(X) | 1 | 1 |
| ENQUEUE(x) | 1 | 3 |
| DEQUEUE(x) | n | 1 |

$\Phi(0) = 0$
We know that if there is nothing to be popped from stack 2, than the element is popped one time and pushed twice. ENQUEUE being 3 ensures that these three times are taken into account. Otherwise, one DEQUEUE being 1 accounts for other situations
This ensures $\Phi(n) - \Phi(0) \geq 0$ for any n

(c) Let $\Phi(D_n) = 2(stack1.size)$, $\Phi(D_0) = 0$, and $\Phi(D_n) - \Phi(D_0) \geq 0$ for any n
In ENQUEUE:
Cost of PUSH(X) = 1

stack1.size = stack1.size + 1
$\Phi\Delta = \Phi\Delta + 2$
Amortized cost = Actual cost + $\Phi\Delta = 3$
So we can conclude the amortized cost for ENQUEUE is 3
In DEQUEUE:
If stack2 != EMPTY:
Actual cost of DEQUEUE is 1 (POP is 1).
$\Phi\Delta = 0$
Amortized cost = Actual cost + $\Phi\Delta = 1$
else:
Actual cost of POP = 2(stack1.size)+1
$\Phi\Delta = -2$(stack1.size)
Amortized cost = actual cost + $\Phi\Delta = 1$
So we can conclude the amortized cost for DEQUEUE is 1

---

**2: A data structure supports the following two operations for a set of integers S:(a) INSERT(S,x), which inserts x into S; and**
**(b) DELETELARGERHALF(S), which deletes the largest floor(S/2) elements from S.**
**You can assume (a) that the algorithm MEDIAN computes the median of an array of n integers in time n and (b) the algorithm PARTITION(the PARTITION we have seen is modified to accept an input pivot) on such an array takes time n.Using the potential method of analysis, prove that it is possible to implement the data structure using an array (with a variable size) such that a sequence of n operations on the data structure will execute in O(n)time.**

---

| Opn | Acutal Cost | Amortized Cost |
|---|---|---|
| INSERT(S,x) | 2 | $2 + 2(c_1 + c_2)$ |
| DELETELARGERHALF(S) | $(c_1 + c_2)\|S\| + 1$ | 1 |

Let $\Phi = 2(c_1 + c_2)|S|$
So we can say $\Phi(D_0) = 2(c_1 + c_2)(0)$
Because $\Phi(D_0) = 0$ we can say that it is less than any $\Phi(D_n)$ for any value of n (because you cannot have a set cannot have a negative number of elements).
We know for INSERT that the Total Amortised cost = total actual cost + $\Phi(D_n) - \Phi(D_0)$ and $\Phi(D_n) = 2(c_1 + c_2)n$
$= 2n + 2(c_1 + c_2)n - 0 = 2n + 2c_1n + 2c_2n$ which gives us O(n)
We know for DELETELARGERHALF that the Total Amortised cost = total actual cost + $\Phi(D_n) - \Phi(D_0)$ and
$= (c_1 + c_2)n + 1 + 2(c_1 + c_2)n - 0 = 3c_1n + 3c_2n + 1$ which gives us O(n)