Problem Set 6

Justin Ely

615.202.81.FA15 Data Structures

13 October, 2015

---

1)

```
# Nodes are initialized to Null left and right pointers
# and can have a value
class Node(value)
    value = value
    left = Null
    right = Null

class Deque:
    # Right and left pointers are Null when empty
    RightNode = Null
    LeftNode = Null

    def isEmpty()
        if RightNode == Null and LeftNode == Null:
            return True
        else
            return False

    def InsertLeft(value)
        # initialized NewNode to null left and right
        NewNode = Node(value)

        if isEmpty()
            RightNode = NewNode
            LeftNode = NewNode
        else
            NewNode.right = LeftNode
            NewNode.left = LeftNode.next
            LeftNode.left = NewNode

            LeftNode = NewNode
```

```
    def DeleteRight()
        if isEmpty()
            raise Exception
        else
            # store the node temporarily
            tmp = RightNode

            RightNode.left.right = RightNode.right
            RightNode = RightNode.left

            return tmp.value
```

---

2)

```
# Nodes are initialized to Null left and right pointers
# and can have a value
class Node(value)
    value = value
    left = Null
    right = Null

class Deque:
    # initialize header node with Null pointers
    HeadNode = Node(Null)
    HeadNode.right = HeadNode
    HeadNode.left = HeadNode

    # initialize Left and Right sides of deque
    RightNode = Header.right
    LeftNode = Header.left

    def isEmpty()
        if RightNode.left == LeftNode.right:
            return True
        else
            return False

    def InsertRight(value)
        # initialized NewNode to null left and right
        NewNode = Node(value)
```

```
        RightNode.right = NewNode
        NewNode.left = RightNode

        LeftNode.left = NewNode
        NewNode.right = LeftNode

        RightNode = NewNode

    def DeleteLeft()
        if isEmpty()
            raise Exception
        else
            tmp = LeftNode
            LeftNode.right.left = LeftNode.left
            LeftNode = LeftNode.right
            RightNode.right = LeftNode

            return tmp.value
```

---

3)

```
#  An array data structure that handles
#  the sharing of resources
class SharedArray(N_NODES)
    # array of nodes with (value, next)
    data = Array(N_NODES)

    FreeIndx = Stack()

    for (i=0; i<N_NODES; i++)
        FreeIndex.push(i)

    def Borrow():
        if FreeIndx.isEmpty():
            raise Excetion

        return FreeIndx.pop()

    def Return(index):
        FreeIndx.push(index)
```

```
//----------------------------------------

# Stack takes as a SharedArray instance upon instantiation
class FriendlyStack(Sdata)
    StartNode = -1

    def isEmpty()
        if StartNode == -1
            return True
        else
            return False

    def Push(value)
        newIndx = Sdata.Borrow()

        if startNode == -1:
            startNode = newIndx
        Sdata.data[newIndx].value = value
        Sdata.data[newIndx].next = -1

    def Pop()
        if isEmpty()
            raise Exception

        node = StartNode
        while not node.next == -1:
            prev = node
            node = data[node.next]

        data[prev].next = -1
        data.Return(node)

        return Sdata.data[node]

//----------------------------------------

# Queue takes as a SharedArray instance upon instantiation
class FriendlyQueue(Sdatae)
    StartNode = -1

    def isEmpty()
        if StartNode == -1
            return True
```

```
    else
        return False

def Insert(value)
    newIndx = Sdata.Borrow()

    if startNode == -1:
        startNode = newIndx
    Sdata.data[newIndx].value = value
    Sdata.data[newIndx].next = -1

def Delete
     if isEmpty()
        raise Exception

    tmp = StartNode
    StartNode = Sdata.data[StartNode.next]

    Sdata.Return(tmp)
    return Sdata.data[tmp]
```