**Doubly Linked List implementation**

**Code:**

class node:

    def \_\_init\_\_(self, e):

        self.element = e

        self.next = None

        self.prev = None

class DLList:

    def \_\_init\_\_(self):

        self.head = None

        self.sz = 0

    def insertLast(self,e):

        newNode = node(e)

        if self.isEmpty():

            self.head = newNode

        else:

            temp = self.head

            while temp.next != None:

                temp = temp.next

            temp.next = newNode

            newNode.prev = temp

        self.sz += 1

        return

    def insertFirst(self,e):

        newNode = node(e)

        if self.isEmpty():

            self.head = newNode

        else:

            temp = self.head

            newNode.next = temp

            temp.prev = newNode

            self.head = newNode

        self.sz += 1

        return

    def deleteFirst(self):

        if not self.isEmpty():

            temp = self.head

            self.head = temp.next

            self.head.prev = None

            del temp

            self.sz -= 1

        return

    def deleteLast(self):

        if not self.isEmpty():

            temp = self.head

            while temp.next != None:

                temp = temp.next

            temp = temp.prev

            del temp.next

            temp.next = None

            self.sz -= 1

        return

    def printListForward(self):

        tnode = self.head

        while tnode!= None:

            print(tnode.element,end=" ")

            tnode = tnode.next

        print("")

        return

    def printListBackward(self):

        tnode = self.tail

        while tnode!= None:

            print(tnode.element,end=" ")

            tnode = tnode.prev

        print("")

        return

    def findNode(self, val):

        curnode = self.head

        while curnode!=None:

            if curnode.element == val:

                return curnode

            curnode = curnode.next

        return None

    def isEmpty(self):

        if self.sz == 0:

            return True

        else:

            return False

    def size(self):

        return self.sz

def testDLL():

    dll = DLList()

    inputs=int(input())

    while inputs>0:

        command=input()

        operation=command.split()

        if(operation[0]=="S"):

            print(dll.size())

        elif(operation[0]=="I"):

            print(dll.isEmpty())

        elif(operation[0]=="IF"):

            dll.insertFirst(int(operation[1]))

            dll.printListForward()

        elif(operation[0]=="IL"):

            dll.insertLast(int(operation[1]))

            dll.printListForward()

        elif(operation[0]=="DF"):

            dll.deleteFirst()

            dll.printListForward()

        elif(operation[0]=="DL"):

            dll.deleteLast()

            dll.printListForward()

        inputs-=1

def main():

    testDLL()

if \_\_name\_\_ == '\_\_main\_\_':

    main()

**output:**

**5**

**IF 5**

**5**

**IF 4**

**4 5**

**DF**

**5**

**IL 4**

**5 4**

**DL**

**5**