class CircularArray:

    def \_\_init\_\_(self, lin, st, sz):

        # Initializing Variables

        self.start = st

        self.size = sz

        self.lin=lin

        self.cir=[None]\*len(self.lin)

        x=self.start

        for i in range(self.size):

            self.cir[x]=self.lin[i]

            x=(x+1)%len(self.lin)

    def printFullLinear(self):

        for i in range(len(self.cir)):

            if i !=len(self.cir)-1:

                print(self.cir[i],end=', ')

            else:

                print(self.cir[i])

    def printForward(self):

        x=self.start

        for i in range(self.size):

            if i !=self.size-1:

                print(self.cir[x],end=', ')

            else:

                print(self.cir[x])

            x=(x+1)%len(self.cir)

    def printBackward(self):

        x=(self.start+self.size-1)%len(self.cir)

        for i in range(self.size):

            if i !=self.size-1:

                print(self.cir[x],end=', ')

            else:

                print(self.cir[x])

            x-=1

            if x==-1:

                x=len(self.cir)-1

    def linearize(self):

        x=self.start

        y=[None]\*self.size

        for i in range(self.size):

            y[i]=self.cir[x]

            x=(x+1)%len(self.cir)

        self.cir=y

    def resizeStartUnchanged(self,a):

        y=[None]\*a

        x=self.start

        z=self.start

        for i in range(self.size):

            y[x]=self.cir[z]

            x=(x+1)%a

            z=(z+1)%len(self.cir)

        self.cir=y

        self.size=a

    def palindromeCheck(self):

        x=(self.start+self.size-1)%len(self.cir)

        y=self.start

        z=True

        for i in range(self.size//2):

            if self.cir[y]!=self.cir[x]:

                z=False

            x-=1

            y+=1

            if x==-1:

                x=len(self.cir)-1

        if z==True:

            print('This array is a palindrome')

        else:

            print('This array is NOT a palindrome')

    def sort(self):

        numbers=self.cir

        i=self.start

        for index1 in range(0, self.size-1):

            x=self.cir[i]

            j=(i+1)%len(self.cir)

            index2=index1+1

            while index2!=self.size:

                y=self.cir[j]

                if x>y:

                    self.cir[i]=y

                    self.cir[j]=x

                index2+=1

                j=(j+1)%len(self.cir)

            i=(i+1)%len(self.cir)

    def equivalent(self, cir\_arr):

        x=True

        if self.size==cir\_arr.size:

            y=self.start

            z=cir\_arr.start

            for i in range(self.size):

                if self.cir[y]!=cir\_arr.cir[z]:

                    x=False

                y=(y+1)%len(self.cir)

                z=(z+1)%len(cir\_arr.cir)

        else:

            x=False

        return x

    def intersection(self, c2):

        x=self.start

        count=0

        for i in range(self.size):

            y=c2.start

            for j in range(c2.size):

                if self.cir[x]==c2.cir[y]:

                    count+=1

                y=(y+1)%len(c2.cir)

            x=(x+1)%len(self.cir)

        z=[None]\*count

        k=0

        x=self.start

        for i in range(self.size):

            y=c2.start

            for j in range(c2.size):

                if self.cir[x]==c2.cir[y]:

                    z[k]=self.cir[x]

                    k+=1

                y=(y+1)%len(c2.cir)

            x=(x+1)%len(self.cir)

        return z

lin\_arr1 = [10, 20, 30, 40, None]

print("==========Test 1==========")

c1 = CircularArray(lin\_arr1, 2, 4)

c1.printFullLinear()

c1.printForward()

c1.printBackward()

print("==========Test 2==========")

c1.linearize()

c1.printFullLinear()

print("==========Test 3==========")

lin\_arr2 = [10, 20, 30, 40, 50]

c2 = CircularArray(lin\_arr2, 2, 5)

c2.printFullLinear()

c2.resizeStartUnchanged(8)

c2.printFullLinear()

print("==========Test 4==========")

lin\_arr3 = [10, 20, 30, 20, 10, None, None]

c3 = CircularArray(lin\_arr3, 3, 5)

c3.printForward() # This should print: 10, 20, 30, 20, 10

c3.palindromeCheck() # This should print: This array is a palindrome

print("==========Test 5==========")

lin\_arr4 = [10, 20, 30, 20, None, None, None]

c4 = CircularArray(lin\_arr4, 3, 4)

c4.printForward() # This should print: 10, 20, 30, 20

c4.palindromeCheck() # This should print: This array is NOT a palindrome

print("==========Test 6==========")

lin\_arr5 = [10, 20, -30, 20, 50, 30, None]

c5 = CircularArray(lin\_arr5, 5, 6)

c5.printForward() # This should print: 10, 20, -30, 20, 50, 30

c5.sort()

c5.printForward() # This should print: -30, 10, 20, 20, 30, 50

print("==========Test 7==========")

lin\_arr6 = [10, 20, -30, 20, 50, 30, None]

c6 = CircularArray(lin\_arr6, 2, 6)

c7 = CircularArray(lin\_arr6, 5, 6)

c6.printForward() # This should print: 10, 20, -30, 20, 50, 30

c7.printForward() # This should print: 10, 20, -30, 20, 50, 30

print(c6.equivalent(c7)) # This should print: True

print("==========Test 8==========")

lin\_arr7 = [10, 20, -30, 20, 50, 30, None, None, None]

c8 = CircularArray(lin\_arr7, 8, 6)

c6.printForward() # This should print: 10, 20, -30, 20, 50, 30

c8.printForward() # This should print: 10, 20, -30, 20, 50, 30

print(c6.equivalent(c8)) # This should print: True

print("==========Test 9==========")

lin\_arr8 = [10, 20, 30, 40, 50, 60, None, None, None]

c9 = CircularArray(lin\_arr8, 8, 6)

c6.printForward() # This should print: 10, 20, -30, 20, 50, 30

c9.printForward() # This should print: 10, 20, 30, 40, 50, 60

print(c6.equivalent(c9)) # This should print: False

print("==========Test 10==========")

lin\_arr9 = [10, 20, 30, 40, 50, None, None, None]

c10 = CircularArray(lin\_arr9, 5, 5)

c10.printFullLinear() # This should print: 40, 50, None, None, None, 10, 20, 30

lin\_arr10 = [5, 40, 15, 25, 10, 20, 5, None, None, None, None, None]

c11 = CircularArray(lin\_arr10, 8, 7)

c11.printFullLinear() # This should print: 10, 20, 5, None, None, None, None, None, 5, 40, 15, 25

output = c10.intersection(c11)

print(output) # This should print: [10, 20, 40]