class PriorityQueue:

def \_\_init\_\_(self):

self.pq =[]

def isEmpty(self):

if(len(self.pq) == 0):

return True

return False

def PqSize(self):

return len(self.pq)

def getMin(self):

if(self.isEmpty()):

return 0

return self.pq[0]

def insertInPq(self , element ):

self.pq.append(element)

childInd = len(self.pq)-1

while(childInd > 0):

parentIndex = (childInd-1)//2

if(self.pq[childInd] < self.pq[parentIndex]):

temp = self.pq[childInd]

self.pq[childInd] = self.pq[parentIndex]

self.pq[parentIndex] = temp

else:

break

childInd = parentIndex

def removeMin(self):

if(len(self.pq) < 1):

return 0

if(len(self.pq) == 1):

return self.pq.pop()

minValue = self.pq[0]

self.pq[0] = self.pq[len(self.pq)-1]

self.pq.pop()

currentIndex = 0

left = (currentIndex\*2)+1

right = (currentIndex\*2)+2

while(currentIndex < len(self.pq)):

print(left, " ", right ,end=" ")

if(left < len(self.pq) and right < len(self.pq) and self.pq[left] < self.pq[right] ):

if(self.pq[left] < self.pq[currentIndex]):

temp = self.pq[left]

self.pq[left] = self.pq[currentIndex]

self.pq[currentIndex] = temp

currentIndex = left

else:

break

elif(right < len(self.pq)):

if(self.pq[right] < self.pq[currentIndex]):

temp = self.pq[right]

self.pq[right] = self.pq[currentIndex]

self.pq[currentIndex] = temp

currentIndex = right

else:

break

else:

break

left = (currentIndex\*2)+1

right = (currentIndex\*2)+2

return minValue

def displayPriorityQueue(self):

print(self.pq)

ob = PriorityQueue()

ob.insertInPq(2)

ob.insertInPq(1)

ob.insertInPq(4)

ob.insertInPq(3)

print(ob.getMin())

ob.displayPriorityQueue()

ob.insertInPq(9)

ob.insertInPq(11)

ob.insertInPq(7)

print("size of the PQ : ",ob.PqSize())

print(ob.PqSize())

print(ob.removeMin())

print(ob.removeMin())

print(ob.removeMin())

print(ob.removeMin())

print(ob.removeMin())

print(ob.removeMin())

print(ob.removeMin())

ob.displayPriorityQueue()