Exercise:10

class MinHeap:

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

self.heap = []

def \_parent(self, i):

return (i - 1) // 2

def \_left(self, i):

return 2 \* i + 1

def \_right(self, i):

return 2 \* i + 2

def \_heapify\_up(self, i):

while i > 0 and self.heap[self.\_parent(i)] > self.heap[i]:

self.heap[self.\_parent(i)], self.heap[i] = self.heap[i], self.heap[self.\_parent(i)]

i = self.\_parent(i)

def \_heapify\_down(self, i):

smallest = i

left = self.\_left(i)

right = self.\_right(i)

if left < len(self.heap) and self.heap[left] < self.heap[smallest]:

smallest = left

if right < len(self.heap) and self.heap[right] < self.heap[smallest]:

smallest = right

if smallest != i:

self.heap[i], self.heap[smallest] = self.heap[smallest], self.heap[i]

self.\_heapify\_down(smallest)

def insert(self, val):

self.heap.append(val)

self.\_heapify\_up(len(self.heap) - 1)

def peek(self):

if not self.heap:

return None

return self.heap[0]

def delete(self, val):

try:

index = self.heap.index(val)

self.heap[index] = self.heap[-1]

self.heap.pop()

if index < len(self.heap):

self.\_heapify\_down(index)

self.\_heapify\_up(index)

except ValueError:

print(f"Value {val} not found in heap.")

def display(self):

print(self.heap)

# Example usage

heap = MinHeap()

heap.insert(10)

heap.insert(5)

heap.insert(20)

heap.insert(1)

print("Heap after inserts:")

heap.display()

print("Peek:", heap.peek())

heap.delete(5)

print("Heap after deleting 5:")

heap.display()

heap.delete(heap.peek())

print("Heap after deleting root (min element):")

heap.display()

output:

Heap after inserts:

[1, 5, 20, 10]

Peek: 1

Heap after deleting 5:

[1, 10, 20]

Heap after deleting root (min element):

[10, 20]