Baes, Franz Emmanuel F.
Activity 9 DSA

OUTPUT

```
PS D:\DSA\ACTIVITY 9> & "D:/DSA/ACTIVITY 9/.venv/Scripts/python.exe" "Baes_Act9.py"
==================================================
Testing Min-Heap Implementation
==================================================

--- Test 1: Inserting elements ---
Inserted 10 into the heap
Inserted 5 into the heap
Inserted 20 into the heap
Inserted 1 into the heap
Inserted 15 into the heap
Inserted 30 into the heap
Current heap: [1, 5, 20, 10, 15, 30]

--- Test 2: Get minimum element ---
Minimum element: 1

--- Test 3: Deleting minimum elements ---
Deleted minimum value: 1
Current heap: [5, 10, 20, 30, 15]
Deleted minimum value: 5
Current heap: [10, 15, 20, 30]

--- Test 4: Insert more elements ---
Inserted 3 into the heap
Inserted 8 into the heap
Current heap: [3, 10, 8, 30, 15, 20]

--- Test 5: Heapify from array ---
Heapified array: [25, 10, 35, 5, 15, 40, 30]
Current heap: [5, 10, 30, 25, 15, 40, 35]

--- Test 6: Delete all elements ---
Deleted minimum value: 5
Current heap: [10, 15, 30, 25, 35, 40]
Deleted minimum value: 10
Current heap: [15, 25, 30, 40, 35]
Deleted minimum value: 15
Current heap: [25, 35, 30, 40]
Deleted minimum value: 25
Current heap: [30, 35, 40]
Deleted minimum value: 30
Current heap: [35, 40]
Deleted minimum value: 35
Current heap: [40]
Heap is empty

--- Test 7: Delete from empty heap ---
Heap is empty, cannot delete

==================================================
All tests completed!
==================================================
```

Baes, Franz Emmanuel F.
Activity 9 DSA

Source Code

Baes_Act9.py > MinHeap > rightChild

```python
# Activity 9: Heap Implementation

class MinHeap:
    def __init__(self):
        self.heap = []

    def parent(self, i):
        return (i - 1) // 2

    def leftChild(self, i):
        return 2 * i + 1

    def rightChild(self, i):
        return 2 * i + 2

    def swap(self, i, j):
        self.heap[i], self.heap[j] = self.heap[j], self.heap[i]

    def insert(self, value):
        self.heap.append(value)
        self.heapifyUp(len(self.heap) - 1)
        print(f"Inserted {value} into the heap")

    def heapifyUp(self, i):
        while i > 0 and self.heap[i] < self.heap[self.parent(i)]:
            self.swap(i, self.parent(i))
            i = self.parent(i)

    def deleteMin(self):
        if len(self.heap) == 0:
            print("Heap is empty, cannot delete")
            return None

        if len(self.heap) == 1:
            return self.heap.pop()

        minVal = self.heap[0]
        self.heap[0] = self.heap.pop()
        self.heapifyDown(0)

        print(f"Deleted minimum value: {minVal}")
        return minVal

    def heapifyDown(self, i):
        minIndex = i
        left = self.leftChild(i)
        right = self.rightChild(i)

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

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

        if minIndex != i:
            self.swap(i, minIndex)
            self.heapifyDown(minIndex)
```

Baes_Act9.py > MinHeap > rightChild

```
3   class MinHeap:
58
59      def heapify(self, arr):
60          self.heap = arr.copy()
61          for i in range(len(self.heap) // 2 - 1, -1, -1):
62              self.heapifyDown(i)
63          print(f"Heapified array: {arr}")
64
65      def display(self):
66          if len(self.heap) == 0:
67              print("Heap is empty")
68          else:
69              print(f"Current heap: {self.heap}")
70
71      def getMin(self):
72          if len(self.heap) == 0:
73              return None
74          return self.heap[0]
75
76
77  def testHeap():
78      print("=" * 50)
79      print("Testing Min-Heap Implementation")
80      print("=" * 50)
81
82      heap = MinHeap()
83
84      print("\n--- Test 1: Inserting elements ---")
85      heap.insert(10)
86      heap.insert(5)
87      heap.insert(20)
88      heap.insert(1)
89      heap.insert(15)
90      heap.insert(30)
91      heap.display()
92
93      print("\n--- Test 2: Get minimum element ---")
94      print(f"Minimum element: {heap.getMin()}")
95
96      print("\n--- Test 3: Deleting minimum elements ---")
97      heap.deleteMin()
98      heap.display()
99      heap.deleteMin()
100     heap.display()
101
102     print("\n--- Test 4: Insert more elements ---")
103     heap.insert(3)
104     heap.insert(8)
105     heap.display()
106
107     print("\n--- Test 5: Heapify from array ---")
108     heap2 = MinHeap()
109     testArray = [25, 10, 35, 5, 15, 40, 30]
110     heap2.heapify(testArray)
111     heap2.display()
112
113     print("\n--- Test 6: Delete all elements ---")
114     while heap2.getMin() is not None:
115         heap2.deleteMin()
```

```
114     while heap2.getMin() is not None:
115         heap2.deleteMin()
116         heap2.display()
117
118     print("\n--- Test 7: Delete from empty heap ---")
119     heap2.deleteMin()
120
121     print("\n" + "=" * 50)
122     print("All tests completed!")
123     print("=" * 50)
124
125
126 if __name__ == "__main__":
127     testHeap()
128
```