PROGRAMMING ASSESSMENT

1. Implement a Python class MaxHeap that supports the following operations: insert, delete, and get\_max. Ensure the operations maintain the properties of a max-heap.

Ans : class MaxHeap:

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

self.heap = []

def insert(self, val):

self.heap.append(val)

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

print(f"Inserted {val} into the heap.")

def delete(self):

if len(self.heap) == 0:

print("Error: Cannot delete from an empty heap")

return None

if len(self.heap) == 1:

return self.heap.pop()

root = self.heap[0]

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

self.\_heapify\_down(0)

print(f"Deleted {root} from the heap.")

return root

def get\_max(self):

if len(self.heap) == 0:

print("Error: Cannot get max from an empty heap")

return None

return self.heap[0]

def is\_empty(self):

return len(self.heap) == 0

def \_heapify\_up(self, index):

parent = (index - 1) // 2

if index > 0 and self.heap[index] > self.heap[parent]:

self.heap[index], self.heap[parent] = self.heap[parent], self.heap[index]

self.\_heapify\_up(parent)

def \_heapify\_down(self, index):

largest = index

left\_child = 2 \* index + 1

right\_child = 2 \* index + 2

if left\_child < len(self.heap) and self.heap[left\_child] > self.heap[largest]:

largest = left\_child

if right\_child < len(self.heap) and self.heap[right\_child] > self.heap[largest]:

largest = right\_child

if largest != index:

self.heap[index], self.heap[largest] = self.heap[largest], self.heap[index]

self.\_heapify\_down(largest)

def show\_heap(heap):

if not heap:

print("The Heap is empty[].")

else:

print("Current Heap elements:", heap)

def main():

heap = MaxHeap()

while True:

print("\nMaxHeap Operations:")

print("1. Insert")

print("2. Delete")

print("3. Get Max")

print("4. Show Heap")

print("5. Exit")

choice = input("Enter your choice (1-5): ")

if choice == '1':

val = int(input("Enter the value to insert: "))

heap.insert(val)

elif choice == '2':

deleted\_value = heap.delete()

if deleted\_value is not None:

print(f"Deleted value: {deleted\_value}")

elif choice == '3':

max\_value = heap.get\_max()

if max\_value is not None:

print(f"Max value: {max\_value}")

elif choice == '4':

show\_heap(heap.heap) # Pass the internal heap list

elif choice == '5':

print("Exiting...")

break

else:

print("Invalid choice. Please enter a number between 1 and 5.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

==========================================



 

 