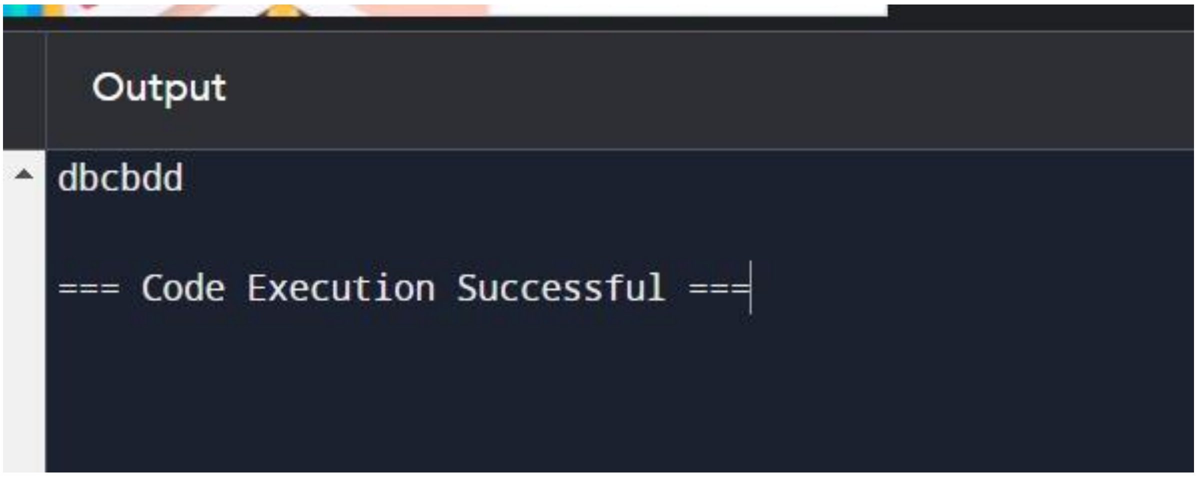
202: Given a list of item weights and the maximum capacity of a container, determine the maximum weight that can be loaded into the container using a greedy approach. The greedy approach should prioritize loading heavier items first until the container reaches its capacity.

Program:

```
from heapq import heappush, heappop, heapify

class Node:

   def _init_(self, char, freq):

      self.char = char

      self.freq = freq

      self.left = None

      self.right = None

def build_huffman_tree(characters, frequencies):

   nodes = []

   for i in range(len(characters)):

      nodes.append(Node(characters[i], frequencies[i]))

   while len(nodes) > 1:

      nodes = sorted(nodes, key=lambda x: x.freq)

      left = nodes.pop(0)

      right = nodes.pop(0)

      parent = Node(None, left.freq + right.freq)

      parent.left = left

      parent.right = right

      nodes.append(parent)

   return nodes[0]

def huffman_decode(root, encoded_string):

   decoded_string = ""

   current = root
```

```
    for bit in encoded_string:
        if bit == '0':
            current = current.left
        else:
            current = current.right
        if current.char:
            decoded_string += current.char
            current = root
    return decoded_string
# Test Case 1
characters1 = ['a', 'b', 'c', 'd']
frequencies1 = [5, 9, 12, 13]
encoded_string1 = '1101100111110'
root1 = build_huffman_tree(characters1, frequencies1)
decoded_message1 = huffman_decode(root1, encoded_string1)
print(decoded_message1)
```

Output:



```
dbcbdd

=== Code Execution Successful ===
```

Time complexity: O(n)