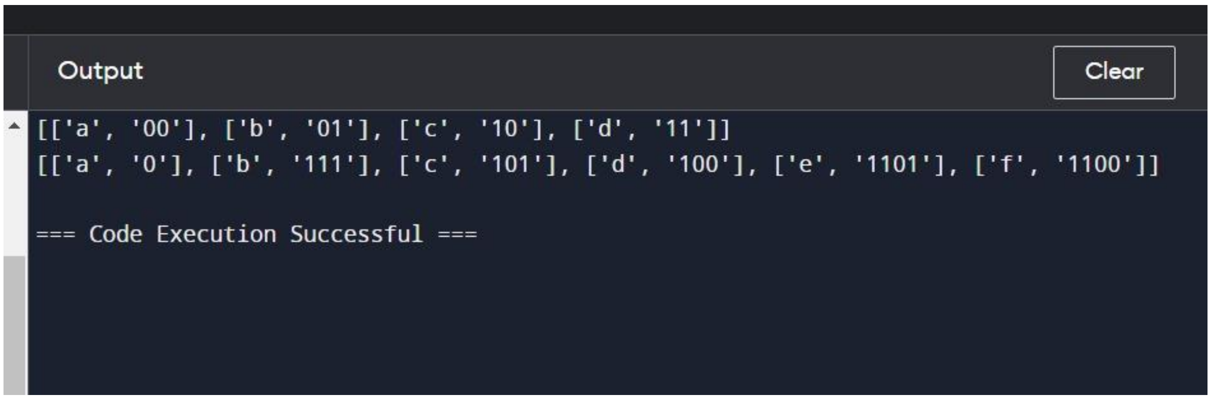
200. Given a set of characters and their corresponding frequencies, construct the Huffman Tree and generate the Huffman Codes for each character.

Program:

```
from heapq import heappush, heappop, heapify

from collections import defaultdict

def huffman_codes(characters, frequencies):

    heap = [[freq, [char, ""]] for char, freq in zip(characters, frequencies)]

    heapify(heap)

    while len(heap) > 1:

        lo = heappop(heap)

        hi = heappop(heap)

        for pair in lo[1:]:

            pair[1] = '0' + pair[1]

        for pair in hi[1:]:

            pair[1] = '1' + pair[1]

        heappush(heap, [lo[0] + hi[0]] + lo[1:] + hi[1:])

    return sorted(heappop(heap)[1:], key=lambda p: (len(p[-1]), p))

# Test Case 1

characters1 = ['a', 'b', 'c', 'd']

frequencies1 = [5, 9, 12, 13]

output1 = huffman_codes(characters1, frequencies1)

print(output1)

# Test Case 2

characters2 = ['f', 'e', 'd', 'c', 'b', 'a']

frequencies2 = [5, 9, 12, 13, 16, 45]

output2 = huffman_codes(characters2, frequencies2)

print(output2)
```

output:



```
[['a', '00'], ['b', '01'], ['c', '10'], ['d', '11']]
[['a', '0'], ['b', '111'], ['c', '101'], ['d', '100'], ['e', '1101'], ['f', '1100']]

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

Time complexity: O(nlogn).