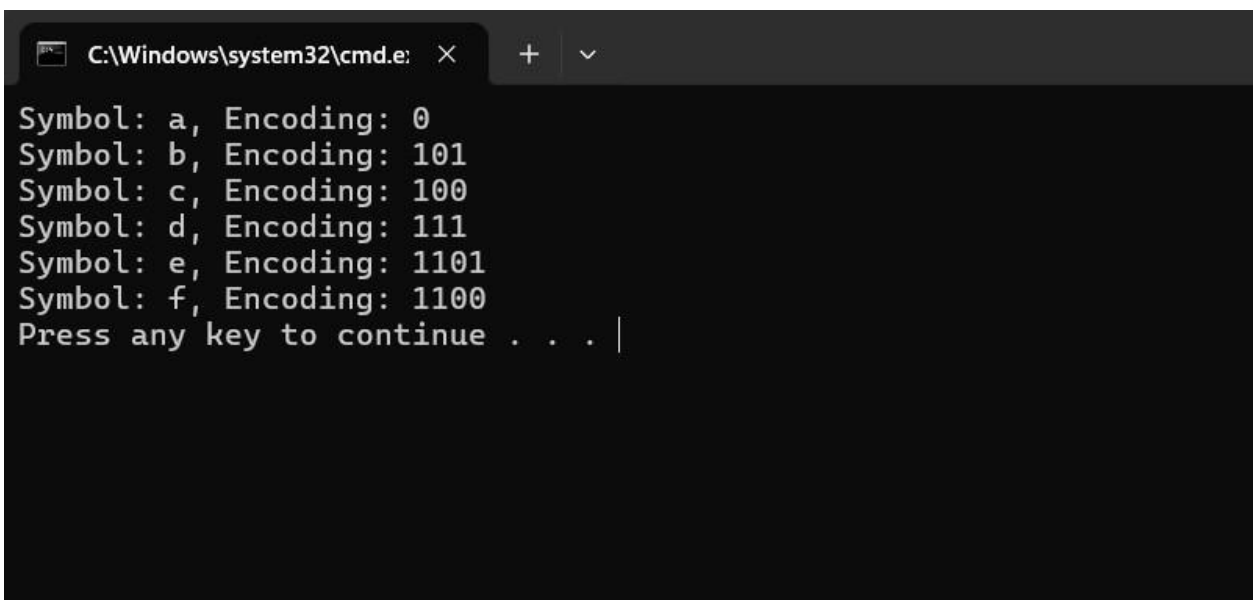
92) Optimal Tree Problem: Huffman Trees and Codes
CODE:

```
from heapq import heappush, heappop, heapify from
collections import defaultdict
 def huffman_tree(symbols):
    h = [[weight, [symbol, ""]]
 for symbol, weight in symbols.items()]
    heapify(h)
    while len(h) > 1:
       lo = heappop(h
)        hi = heappop(h)
        for pair in lo[1:]:
           pair[1] = '0' + pair[1]
        for pair in hi[1:]:
           pair[1] = '1' + pair[1]
       heappush(h, [lo[0] + hi[0]] + lo[1:] + hi[1:])
    return sorted(heappop(h)[1:], key=lambda p: (len(p[-1]), p))

symbols = {'a': 45, 'b': 13, 'c': 12, 'd': 16, 'e': 9, 'f': 5}

 huffman = huffman_tree(symbols)

 for symbol, encoding in huffman:

    print(f'Symbol: {symbol}, Encoding: {encoding}')
```

OUTPUT:



TIME COMPLEXITY : O(nlogn)