In [ ]:
```
# Name: Sushil Suresh Kannake
# Roll no: COBA099
# SUB : DAA
```

In [11]:
```
import heapq

# Creating Huffman tree node
class Node:
    def __init__(self, freq, symbol, left=None, right=None):
        self.freq = freq  # frequency of symbol
        self.symbol = symbol  # symbol name (character)
        self.left = left  # node left of the current node
        self.right = right  # node right of the current node
        self.huff = ''  # tree direction (0/1)

    def __lt__(self, nxt):  # Check if current frequency is less than the next node's freq
        return self.freq < nxt.freq

    def print_nodes(self, val=''):
        new_val = val + str(self.huff)
        # if the node is not an edge node then traverse inside it
        if self.left:
            self.left.print_nodes(new_val)
        if self.right:
            self.right.print_nodes(new_val)

        # if the node is an edge node then display its Huffman code
        if not self.left and not self.right:
            print("{} -> {}".format(self.symbol, new_val))

if __name__ == "__main__":
    chars = ['a', 'b', 'c', 'd', 'e', 'f']
    freq = [5, 9, 12, 13, 16, 45]
    nodes = []

    for i in range(len(chars)):  # converting characters and frequencies into Huffman tree node
        heapq.heappush(nodes, Node(freq[i], chars[i]))

    while len(nodes) > 1:
        left = heapq.heappop(nodes)
        right = heapq.heappop(nodes)

        left.huff = 0
        right.huff = 1
        # Combining the two smallest nodes to create a new node as their parent
        new_node = Node(left.freq + right.freq, left.symbol + right.symbol, left, right)
        # node(freq, symbol, left, right)
        heapq.heappush(nodes, new_node)

    root_node = nodes[0]  # Get the root of the Huffman Tree
    root_node.print_nodes()  # Print Huffman codes
```

```
f -> 0
c -> 100
d -> 101
a -> 1100
b -> 1101
e -> 111
```