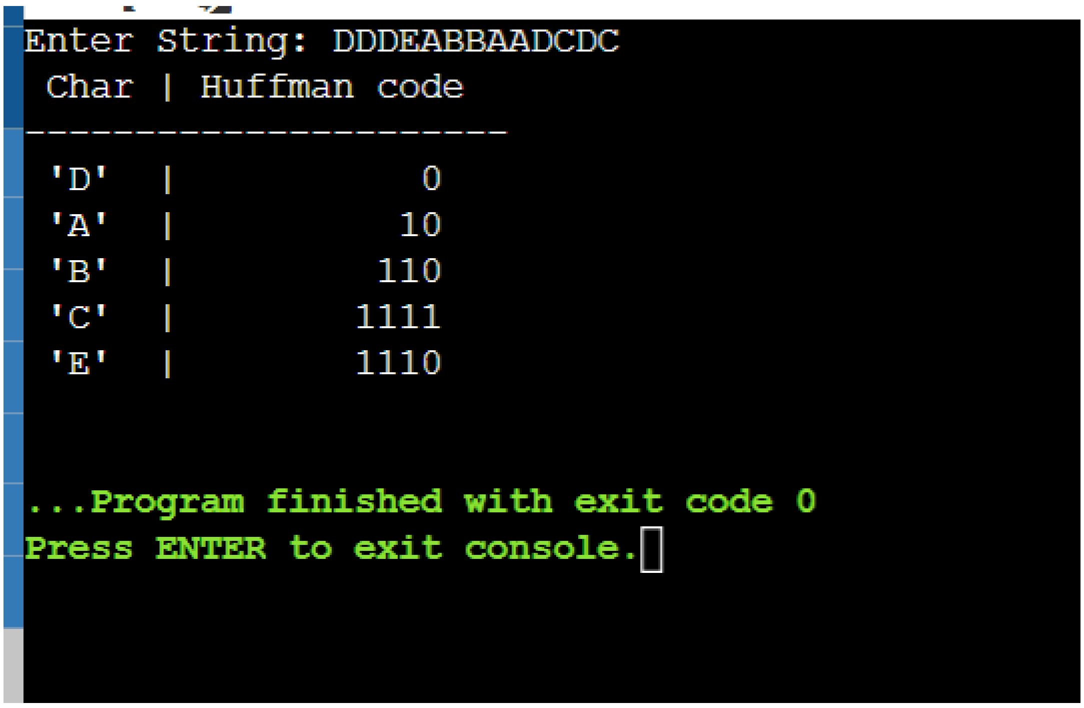
Problem Statement: Write a program to implement Huffman Encoding using a greedy strategy.

CODE:

```
string = input("Enter String: ")
class NodeTree(object):


    def __init__(self, left=None, right=None):
        self.left = left
        self.right = right


    def children(self):
        return (self.left, self.right)


    def nodes(self):
        return (self.left, self.right)


    def __str__(self):
        return '%s_%s' % (self.left, self.right)



def huffman_code_tree(node, left=True, binString=''):
    if type(node) is str:
        return {node: binString}
    (l, r) = node.children()
    d = dict()
    d.update(huffman_code_tree(l, True, binString + '0'))
    d.update(huffman_code_tree(r, False, binString + '1'))
    return d



freq = {}
for c in string:
    if c in freq:
        freq[c] += 1
    else:
        freq[c] = 1


freq = sorted(freq.items(), key=lambda x: x[1], reverse=True)
nodes = freq
while len(nodes) > 1:
    (key1, c1) = nodes[-1]
    (key2, c2) = nodes[-2]
    nodes = nodes[:-2]
    node = NodeTree(key1, key2)
    nodes.append((node, c1 + c2))


    nodes = sorted(nodes, key=lambda x: x[1], reverse=True)
huffmanCode = huffman_code_tree(nodes[0][0])
print(' Char | Huffman code ')
print('----------------------')
for (char, frequency) in freq:
    print(' %-4r |%12s' % (char, huffmanCode[char]))
```

OUTPUT:



```
Enter String: DDDEABBAADCDC
 Char | Huffman code
----------------------
 'D'  |           0
 'A'  |          10
 'B'  |         110
 'C'  |        1111
 'E'  |        1110


...Program finished with exit code 0
Press ENTER to exit console.
```