Birla Institute of Technology & Science, Pilani
Data Structures & Algorithms (CS F211)
Lab Assignment – 4 (Huffman Encoding)

Instructions:

- All input expressions should be read from stdin and output should be printed on stdout.
- For 1 hour 45 min, only a subset of test cases will be visible to students after submitting the code on the portal. After 1hour 45 min, all test cases will be made visible and they will have last 15 min to correct their code and resubmit.
- At the end of 2 hour period, the online system will stop accepting the submissions.
- Only the last submission by the student before end of lab will be considered for evaluation.
- Following messages by online portal will **tentatively** fetch these marks:
  - Correct ➔ 4 marks
  - Wrong-answer (correct for more than half test cases) ➔ 3 marks
  - Run-error/Compiler-error/Timelimit-error ➔ 2 marks
- All submitted source code will be later checked manually by the instructor and final marks will be awarded, which will be posted on Nalanda after the lab assignment has been done by all lab sections.
- Solution must be implemented using the algorithm and data structures mentioned in the lab sheet only.

Problem

Encoding of a message using Huffman Encoding

Input

You will be given a text message as input, consisting of upper case characters only

Output

You will have to output following four strings (each in a new line):

1. No. of nodes in the Huffman tree
2. Post order traversal of the Huffman tree.
   a. For non-leaf nodes, you should print "0" (zero).
   b. For leaf nodes, you should print the corresponding character.
3. In order traversal of Huffman tree
4. Binary Encoding (using the Huffman Code) of the text message

Procedure

1. Read the input and make the frequency table of the characters. Frequency table should be populated in the order of first occurrence of characters in input string i.e. if input is QPPQPR then first entry in frequency table should be of Q.
2. Create a Min-Priority-Queue using a Min-Heap. You will have to implement the following functions: Min-Heapify, Build-Min-Heap, Heap-Extract–Min, Heap-Decrease-Key, and Min-Heap-Insert.
3. Using the frequency table, initialize the Min-Priority-Queue by using the frequency as the key. Make sure that First-in-first-out (FIFO) order is ensured between the elements of same frequency.
4. Create the Huffman Tree using the Huffman's algorithm:

```
while (there are at least two nodes in the priority queue)
{
Let n1 be the first deleted node from the priority queue
Let n2 be the second deleted node from the priority queue
Let n be a new node.
Set the pointers of n, n1 & n2 so that n1 is left child of n and
   n2 is right child of n.
Set the key of n as n.key=n1.key + n2.key.
Insert the node n into the priority queue
}
```

5. Count the number of nodes in the Huffman tree and print the first part of output.
6. Perform a post order traversal of the Huffman tree and print the second part of output.
7. Scan the input message once again, one character at a time printing the Huffman Code corresponding to the character. For printing the Huffman Code corresponding to the character, you will have to start from the leaf node corresponding to the character and move up towards the root.

The Code will be the binary string in reverse (from root to the leaf node). While moving to a left child, print 0 and while moving to a right child, print 1.

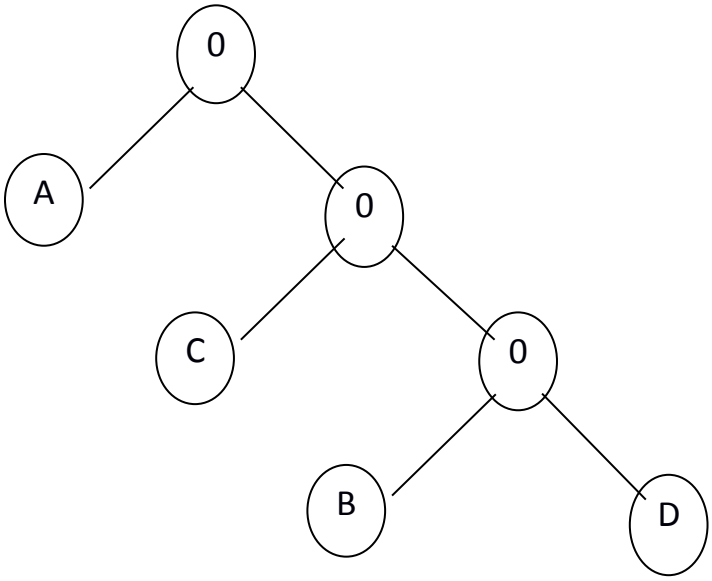
Sample Test case
Input:

ABACCDA

Output:

7
ACBD000
A0C0B0D
0110010101110

Explanation
The Huffman Tree will be:



Postorder Traversal: ACBD000

In order traversal: A0C0B0D

Huffman Codes:

A – 0
B – 110
C – 10
D – 111