CODING ASSIGNMENT

Name : Md Muktadir Mzumder
Id : 190042136
Department : Software Engineering

Course title : Discrete Mathematics
Course code: CSE 4203

3. Illustrate in a typed report, the sequence of generating Prefix Codes using Huffman Coding for the same string just to verify the output of your program in Question-2

REPORT WRITTING

Report Writing on Sequence of Generating Prefix Code Using Huffman Coding

Introduction:

The Coding Assignment "Generating Prefix Codes using Huffman Coding" is implemented using programming language python.Huffman Coding In computer science and information theory, a Huffman code is a particular type of optimal prefix code that is commonly used for lossless data compression.The output from Huffman's algorithm can be viewed as a variable-length code table for encoding a source symbol (such as a character in a file).
In this coding task, using Huffman Coding Prefix Codes of the letters of a string is generated.

Report on the verification of the output of the sequence of generating Prefix Codes using Huffman Coding for the particular string , taken for the task is illustrated below:

TASK 1:

In this first task, a function is basically created that will generate a string by considering the last 4-digits of a Student ID.The input that will be given by the user will be his or her Student-Id. And that to store that input; a variable is used of data-type,String.

```
## Function: for converting number to string

def numbertowords(x):
    ## Number dictionary ##
    s = {0: 'ZERO', 1: 'ONE', 2: 'TWO', 3: 'THREE', 4: 'FOUR', 5: 'FIVE', 6: 'SIX', 7: 'SEVEN', 8: 'EIGHT', 9: 'NINE'}

    ## Last four digits of the given ID
    a = x[-4:]
    string = ''

    ## Convert the last four digits to their corresponding string form
    for i in a:
        d = int(i)
        if d in s:
            string += s[d]

    return 'MYIDIS' + string
```

Fig-1 : User-defined function "numbertowords"

In this user-defined function; for loop is used to generate the string that passed as argument from the Input Section.

```
## Taking input from the user
stuId = input("Step #1. Enter Student ID:")

## Calling function to convert the ID to string
IdStr = numbertowords(stuId)
print("Step #2. Generated String: " + IdStr)
```

Fig-2: Input section that calling the "numbertowords" functions

The input data is storing in variable stuId, which of data type, String. And this input will pass to the function "numbertowords" as argument by calling that function . And after the generating the code inside, the called function named numbertowords" the result will be return as String. And the return String will be stored in variable IdStr which is also of data-type String.

Output of Task-1.

```
Step #1. Enter Student ID:190042136
Step #2. Generated String: MYIDISTWOONETHREESIX
```

Fig-3: Output of first Task that generates a string considering the last 4-digits of Student Id

Task 2.

In this part, a user-defined function is made that uses the Huffman Coding to generate the Prefix Codes of the letters in the string "*MYIDISTWOONETHREESIX*"; that is gained in task 1.

The function below is particularly defined for computing the frequency of the letters in the string. Spaces in the string are completely ignored.

```
## Function: for computing the Huffman prefix code

def PrefixCode(IdStr):
    ## Computing the frequency initialization with zero.
    freq = {}
    for ch in IdStr:
        if ch in freq:
            freq[ch] += 1
        else:
            freq[ch] = 1
```

Fig-4 : Function for generating frequency of the letters

Here, in this section, a user-defined function named "PrefixCode" is defined in order to generate the Huffman prefix codes. This function's parameter "IdStr" received the string.

```
PrefixCode(IdStr)
```

Fig-5: User-defined function "PrefixCode"

```
{'M': 1}
{'M': 1, 'Y': 1}
{'M': 1, 'Y': 1, 'I': 1}
{'M': 1, 'Y': 1, 'I': 1, 'D': 1}
{'M': 1, 'Y': 1, 'I': 2, 'D': 1, 'S': 1}
{'M': 1, 'Y': 1, 'I': 2, 'D': 1, 'S': 1, 'T': 1}
{'M': 1, 'Y': 1, 'I': 2, 'D': 1, 'S': 1, 'T': 1, 'W': 1}
{'M': 1, 'Y': 1, 'I': 2, 'D': 1, 'S': 1, 'T': 1, 'W': 1, 'O': 1}
{'M': 1, 'Y': 1, 'I': 2, 'D': 1, 'S': 1, 'T': 1, 'W': 1, 'O': 2, 'N': 1}
{'M': 1, 'Y': 1, 'I': 2, 'D': 1, 'S': 1, 'T': 1, 'W': 1, 'O': 2, 'N': 1, 'E': 1}
{'M': 1, 'Y': 1, 'I': 2, 'D': 1, 'S': 1, 'T': 2, 'W': 1, 'O': 2, 'N': 1, 'E': 1, 'H': 1}
{'M': 1, 'Y': 1, 'I': 2, 'D': 1, 'S': 1, 'T': 2, 'W': 1, 'O': 2, 'N': 1, 'E': 1, 'H': 1, 'R': 1}
{'M': 1, 'Y': 1, 'I': 3, 'D': 1, 'S': 2, 'T': 2, 'W': 1, 'O': 2, 'N': 1, 'E': 3, 'H': 1, 'R': 1, 'X': 1}
```

Fig-6: Computing the frequency

The frequency of each letter in the string "*MYIDISTWOONETHREESIX*" are given below-

M = 1 Y = 1 I = 3	D = 1 S =2 T = 2	W = 1 O = 2 N = 1	E = 3 H =1 R = 1 X = 1

Table: Generated Frequency

In the function PrefixCode computation of the frequency of the letters in the string is

generated.After the end of this section, the frequency that are found, are sorted in descending order.

```
        ## Sorting the frequency in Descending order
freq = sorted(freq.items(), key=lambda x: x[1], reverse=True)
lenId = len(IdStr)
```

Fig-7: Sorting the frequencies in descending order and calculating length

Later on probabilities are calculated for generating the binary trees. For the temporary prefix codes, both left and right nodes have existing trees. In generating the binary tress, left and right nodes as well as child and parent nodes are determined.

```
## Merging all the prefix codes from the left and right sides of the final binary trees
for i in range(2):
    for j in range(len(huffmanPreCode[i])):
        completeCode[count] = huffmanPreCode[i][j]
        count += 1

## Sorting all the prefix codes in ascending order based on the length of the codes
completeCode = sorted(completeCode, key=len)

print('Step #3. Prefix Codes (Generated Using Huffman Coding):')

## Displaying the prefix code for the unique chars
for ch in IdStr:
    if ch in charLoc.keys():
        print('{:>10} {:>1} {:<0} {:<5}'.format(ch, ':', '', completeCode[charLoc[ch]]))
        del charLoc[ch]
```

Fig-8: Merging of all the prefix codes of the final binary trees

In case of the root nodes in the final binary tree, 0 is added to the left side and 1 for the right side of the binary tree.Than after merging all prefix codes from left and right sides of the final binary tree; sorting of all prefix codes is carried out in ascending order depending on the length of the codes.

Output of Task -3 :

After sorting, prefix codes are displayed for all the unique characters.The output of the program after generating the prefix codes of the letters in the string; using Huffman Coding:

```
Step #3. Prefix Codes (Generated Using Huffman Coding):
         M :  0000
         Y :  0001
         I :  010
         D :  1100
         S :  111
         T :  0110
         W :  1101
         O :  0111
         N :  1010
         E :  001
         H :  1011
         R :  1000
         X :  1001
```

Fig-9: Generated Prefix Codes

Generated Prefix Codes in Task-3:

M = 0000	W = 1101
Y = 0001	O = 0111
I = 010	N = 1010
D = 1100	E = 001
S = 111	H = 1011
T = 0110	R = 1000 X = 1001

This are the Prefix Codes generated using the Huffman Coding.

----------*END*----------