Assignment-3

Data Structures
(Fall-2023)

Deadline: 17th November 2023

Instructions:

- In case of any kind of plagiarism or dishonesty, Instructors won't be hesitant to give 'F' grade.
- Deadline is final & won't be extended in any case.
- Start assignment right now. You won't be able to complete it on last 2 days :)
- You have to submit two.cpp files named as "sender.cpp" and "reciever.cpp"
- The use of any external libraries such as String, cmath etc. is NOT allowed.
- Late submission will result in zero marks in the assignment. Try to submit the assignment at least one day before the deadline to avoid any issues at the last moment.
- All the submissions will be done on Google classroom.
- You have to add both your solution (.cpp) files in a single folder, compress that folder into a ***.zip*** file and submit that file. Please name the zip file after your roll no (A3_23I-XXXX.zip).

Description:

In this assignment, you will create a sender program and a receiver program for efficient communication between users.

Sender Program:
Imagine you have a piece of text stored in a file. The sender program's job is to read this text and shrink its size by encoding it. Encoding here means assigning shorter codes (bits) to the characters that appear more frequently in the text. For example, if the letter 'A' appears a lot in the text, then rather than storing its ASCII value of "1000001" it might be represented by a shorter code such as "00" or "101" etc. The program will then save this smaller, encoded data into a new file.

Receiver Program:
The receiver program does the opposite. It reads the encoded data from the file created by the sender program. Its task is to transform the output of sender program back to the original text data. Essentially, it reverses the encoding process done by the sender program and shows the original text to the user.
In a nutshell, the sender makes the data smaller by representing it character by less number of bits, and the receiver takes this encoded data and turns it back into the original text so that people can understand the message. This efficient method of communication is essential in various fields like computer science and telecommunications.

Let's delve into the steps of the sender process:

Step 1: Take name of a text file and read its content

Step 2: Determine the frequency of each character.

Step 3: Create separate node for each character based on its frequency. Each node will store a character, its frequency, address of left child, and address of right child. The left and right pointers will be NULL, if a node doesn't have any child.

Step 4: Merge the available nodes iteratively.

1. Pick the two nodes having minimum frequency values
2. Calculate the sum of both the frequencies
3. Store the sum value in a new node
4. Make the two selected nodes as children of this new node. Make the least frequency character as a left child of new node and make second least frequency character as a right child
5. Exclude the left and right nodes from the merging process
6. The new node can be selected for merging now.

Step 5: Continue merging nodes until we are left with only one node.

Step 6: Assign binary codes to characters based on their position in the tree, ensuring shorter codes for more common characters.

Step 7: Encode the text of file using binary codes and store the output in a new text file (output.txt).

Step 8: Store the codes in a text file (codes.txt)

Here is an example for your clarity:

Example:

Step 1: Let's consider the text: "AABBAACDDD".

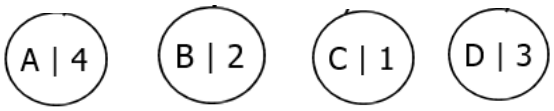
Step 2: Calculate the frequency of each character in the text:

Character	**Frequency**
A	4
B	2
C	1
D	3

Step 3: Create nodes for each character based on their frequency:

Nodes:

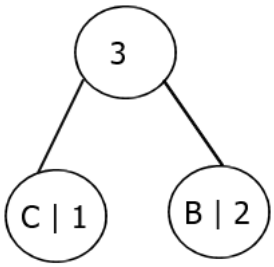
A(4), B(2), C(1), D(3)



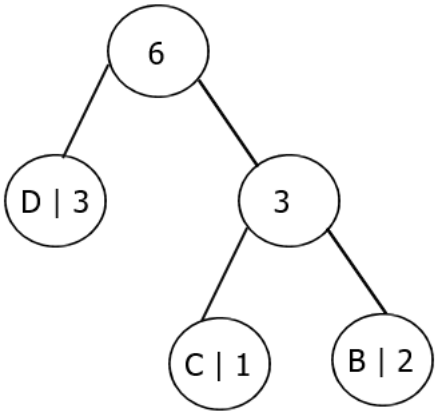
Step 4 & 5: Merge nodes iteratively:

Using this algorithm, the characters can be encoded as follows:

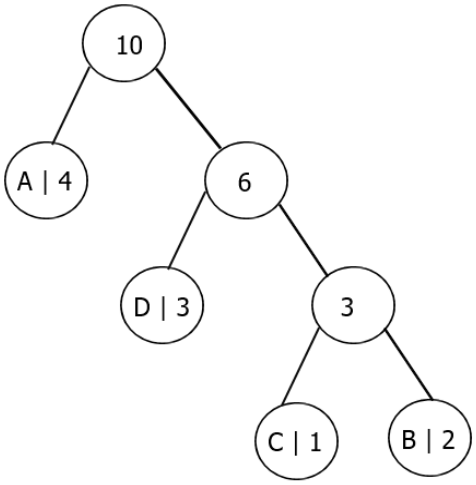
1. Merge C and B: CB (Frequency: 3)



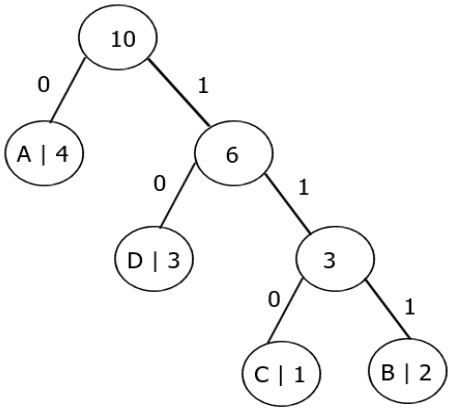
2. Merge CB and D: CBD (Frequency: 6)



3. Merge A and CBD: ACBD (Frequency: 10)



Step 6:
Assign binary codes to characters based on their position in the tree. Assign ‘0’ to left child and ‘1’ to right child on every node.



Codes are as follows:

- A: 0
- D: 10
- C: 110
- B: 111

Step 7: Encode the text of file using binary codes.

AABBAACDDD ->	00111111001101010 10

Store the binary encoded information in “output.txt”

Now, let's delve into the steps of the **receiver** program:

Step 1: Read the “encoded.txt” and “codes.txt”

Step 2: Iterate over encoded text, and find the character against each code”.

Code	0	0	111	111	0	0	110	10	10	10
Letter	A	A	B	B	A	A	C	D	D	D

Step 3: Save the decoded text in “decoded.txt”, and also show it on screen.

Good Luck 😊