**Summer 2021**

**CSC 1310**

**Lab 10: Huffman Compression**

|  |  |
| --- | --- |
| **Reference:** | Chapter 11 – “*11.1 Huffman Compression”* |

**Objective:** To gain a better understanding of Huffman Coding using Priority Queue.

**Source File (s)**: main.cpp, Huffman.h

**INTRODUCTION:**

Given data represented as some quantity of bits, compression transforms the data to use fewer bits. Compressed data uses less storage and can be communicated faster than uncompressed data. The basic idea of compression is to encode frequently-occurring items using fewer bits.

The word “APPLE” has 1 “A”, 1 “E”, 1 “L”,, and 2 “P”(i.e., the frequency A, L, E, and P, are 1, 1, 1, and 2 respectively). Considering that frequently-occurring letters have higher priority, these letters can be compressed by using the Huffman Encoding Algorithm by following steps below:

1. Initially, map each letter with its corresponding frequency.
2. Push all characters which are mapped with corresponding frequency to priority queue (is queue where elements are kept based on priority).
3. To Create Huffman Tree, pop two nodes from priority queue and assign them as left and right child of new node.
4. Sum the priority of two nodes and push the new node formed in priority queue.
5. Repeat all above steps until size of [priority queue](https://www.geeksforgeeks.org/priority-queue-set-1-introduction/) becomes 1.
6. Assign 0 and 1 bit to the left and right child of each node.
7. Traverse from the root to the priority queue for each character and store the[Huffman Code.](https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/)
8. Find the Huffman Code for every character.

The above steps will be clear from the following Huffman tree:

**0** **1**

0 1

0 1

**E 1**

**L 1**

**A 1**

**P 2**

|  |  |
| --- | --- |
| **Characters** | **Codes** |
| P | 0 |
| L | 11 |
| A | 100 |
| E | 101 |

**Draw Huffman tree and find the code of each character in the word “BOOKKEEPER”.**

**WHAT SHOULD THIS PROGRAM DO?**

**Main.cpp file is given to you. Whereas, Huffman.h** are partially provided for you. **You will be completing Huffman.h where you see the following comment section:**

// Write your code here!

**(*Note:* Follow the above steps to complete your Huffman.h file.)**

**How confident are you that your implementation is *correct*? And why?**

**You have finished!**

**In order to receive credit, you must submit this lab exercise (handout) with answers filled in along with the following files:**

* **Huffman.h**
* **main.cpp**

**[*possible full credit*]**