**ZIPPER**

**About Project-**

**Commands in Command shell –**

**>g++ encode.cpp huffman.cpp -o main** -> this commands compiles the programs and create an main.exe file for further use.

**>main inputFile.txt CompressedFile.huf ->** this commands runs the compiled or .exe file of the program along with the arguments.

**>g++ decode.cpp huffman.cpp -o main** -> this commands compiles the programs and create an mainS.exe file for further use.

**>main CompressedFile.huf OutputFile.txt ->** this commands runs the compiled or .exe file of the program along with the arguments.

**encode.cpp & decode.cpp**

Code **- int main(int argc, char\* argv[]) {in encode.cpp & decode.cpp} // line 5**

Int argc – int variable to count number of arguments given

Char\* argv[] – represents the character array that Is given, basically out text file

Here “int argc, char\* argv[]” are command line arguments. We have used them because we are giving the files or running the program through terminal so in such cases we have to use command line arguments. In simple words it is a way of giving input to the program through terminal.

Argc counts the number of argument and argv stores them at 0th index there is programs name and first index Textfile and second index compressed file.

**Flow of code in both programs –**

First the program will check for the number of arguments given that should be 3 such as:

1. Main program.exe file
2. InputTextFile Name (.txt)
3. CompressedFile name (.huf)

IF Program failed to detect these three arguments it will simply throw and error saying “Failed to detect Files“, and exits.

But if arguments are given correctly then,

A function compress/decompress will be called that will perform its task and after completion a message will be generated “Compressed /Decompressed successfully”, along with the file.

**huffman.hpp**

We have created a separate header file that contains all the necessary header files such as

* #include <string> (To work with strings)
* #include <vector> (vectors are dynamic arrays)
* #include <queue> (To use inbuilt priority queue or heaps)
* #include <fstream> (working with file operations like read or write).

//Defining Huffman Tree Node

struct Node {  
 char data; // to store characters of file  
 unsigned freq; // to store frequency of each char  
 string code; // string the code that will tell us position of a char

Node \*left, \*right; // Nodes to create Huffman tree

Node() { // default constructor initialising values to null  
 left = right = NULL;  
 }  
};

Functions in Header File // just like cin, cout, strcpy

priority\_queue <Node\*, vector<Node\*>, Compare> minHeap;

void createArr() - Initializing a vector of tree nodes representing character's ascii value and initializing its frequency with 0

void traverse(Node\*, string);- Traversing the constructed tree to generate huffman codes of each present character

int binToDec(string);- Function to convert binary string to its equivalent decimal value

string decToBin(int);- Function to convert a decimal number to its equivalent binary string

void buildTree(char, string&);- Reconstructing the Huffman tree while Decoding the file

void createMinHeap();- Creating Min Heap of Nodes by frequency of characters in the input file

void createTree();- Constructing the Huffman tree

void createCodes();- Generating Huffman codes

void saveEncodedFile();- Saving Huffman Encoded File

void saveDecodedFile();- Saving Decoded File to obtain the original File

void getTree();- Reading the file to reconstruct the Huffman tree

**huffman.hpp**

In this program we have defined the functionality of the functions that we have defined earlier in the header file.