

**PROJECT REPORT**

**Members:**

1. **Syed Abdul Rehman 21K-3156**
2. **Raja Ibrahim 21K-4585**

**Instructor:**

**Ali Shah Fatmi**

**TA:**

**Tahreem Fatima**

**Title:**

***Huffman Encoding and Decoding (Text & Images)***

**Acknowledgments*:***

First up of all, we would like to thank Sir Ali Shah Fatmi, for his constant support in the

entire semester and in this project, he helped us in different ways, by answering us our queries

and helping us in improving our goals achievement.

**About:**

When storing (most importantly on servers) or transmitting (over a network)

many files, it is highly recommended to compress them. Storing compressed files

saves a lot of space in memory and transmitting such files over a network is

significantly faster. In this project we only focused on compression, using the

Huffman algorithm, of text (.txt) and image (.bmp) files. The reason for choosing

the Huffman algorithm is that it is a lossless compression technique, which means

data will be completely recovered after decompression. In digital communication,

it will design good variable-length codes given the probabilities of the symbols. It

will provide code that’s uniquely decodable and prefix free code by analyzing the

frequencies that certain symbol appears in a text format.

OVERVIEW OF TEXT and IMAGE ENCODING, DECODING:

**Tools and Concepts used:**

We used C++ and for IDE Dev C++ was primarily used. We used the concept of Nodes, classes, operator overloading, filing, recursion, range based for loop, Priority queue to build the Huffman tree and heap data structure to implement it, we also used standard libraries where required. We encoded the text and assigned codes in the form of 0s and 1s to that text, or image. The code was then decoded. In our project we displayed both Compressed and Decompressed i.e., encoded, and decoded.

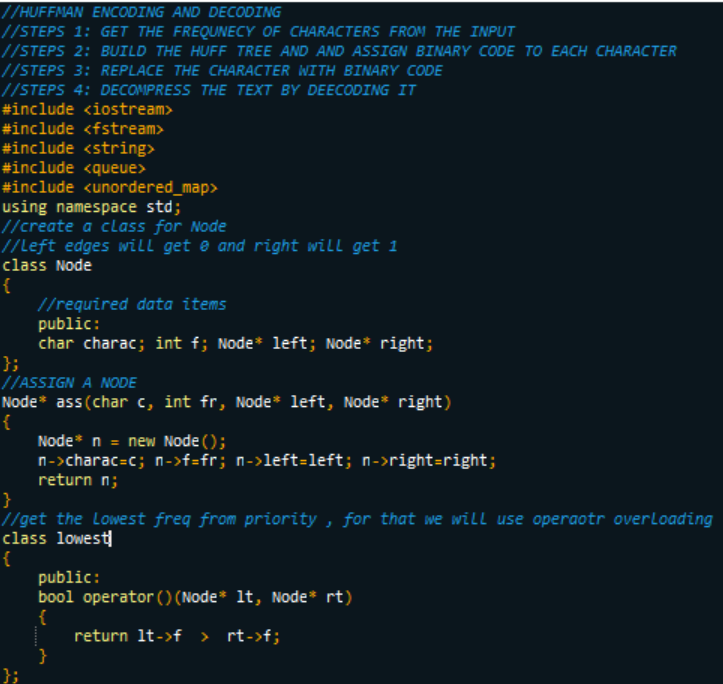
**Use of this project:**

It plays an important role in compressing the text and images files as well, which will be used by digital communication and image processing. It is used in conventional compression formats like GZIP, BZIP2, PKZIP, .BMP files, etc. and in Text and Fax Transmissions.

**Working:**

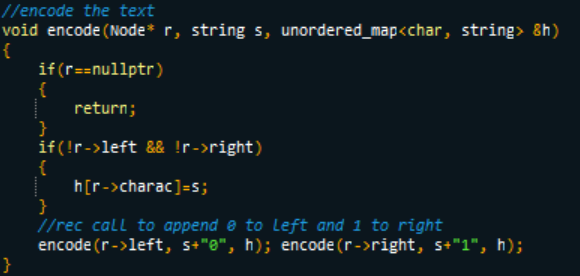
TEXT:

1. Classes:



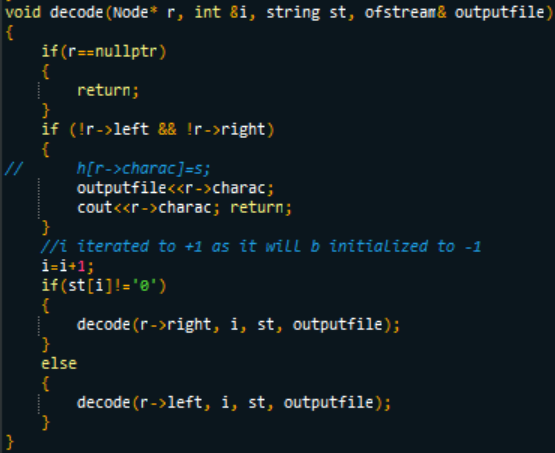
1. Encode Text Function:

The freq of characters is counted, i.e., number of times its occurring, then using priority queue priority is set, lowest freq one having highest priority or vice versa. Then every character is made a leaf node, starting from the smallest one and assigning the minimum frequency to the left child of root node and assigning the second minimum frequency to the right child of root. Root is basically the sum of these two nodes, then removing the two min freq from Q, and add the sum into list, then this goes on for all freq, after that if we are going to count the bits then its reduced.

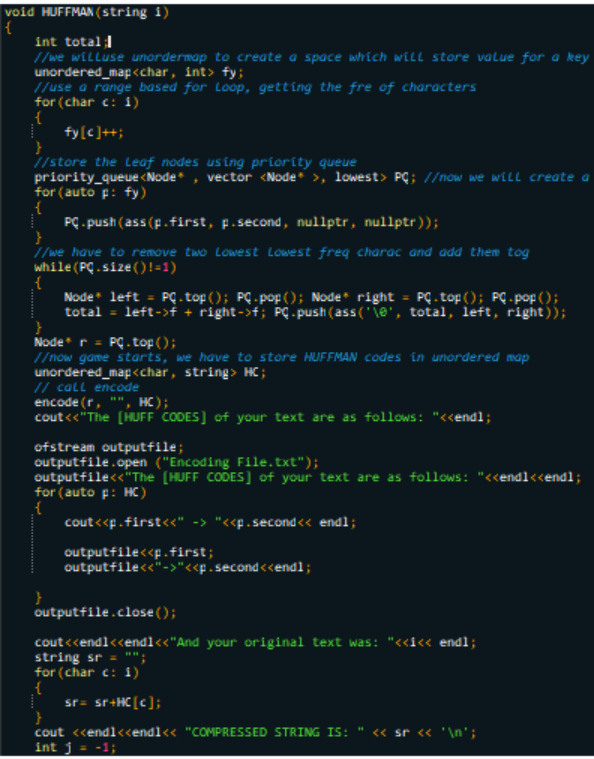


1. Decode Text Function:

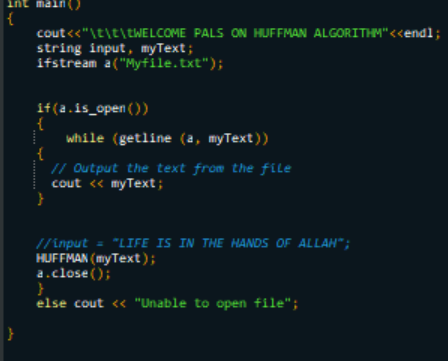
For decoding the code, we can take the code and traverse through the tree to find the character.



1. Main Function of our code:

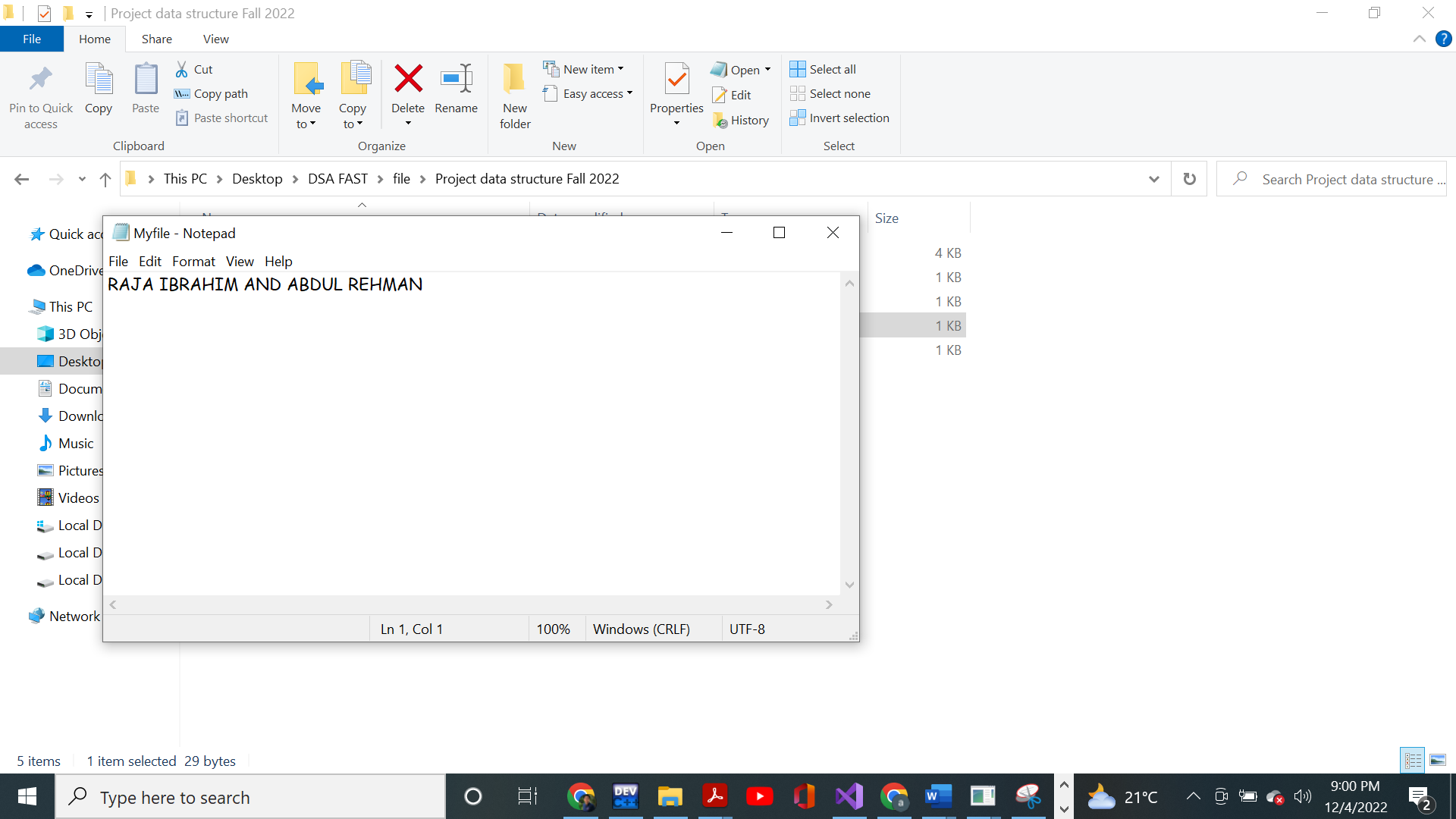


1. Int main ()

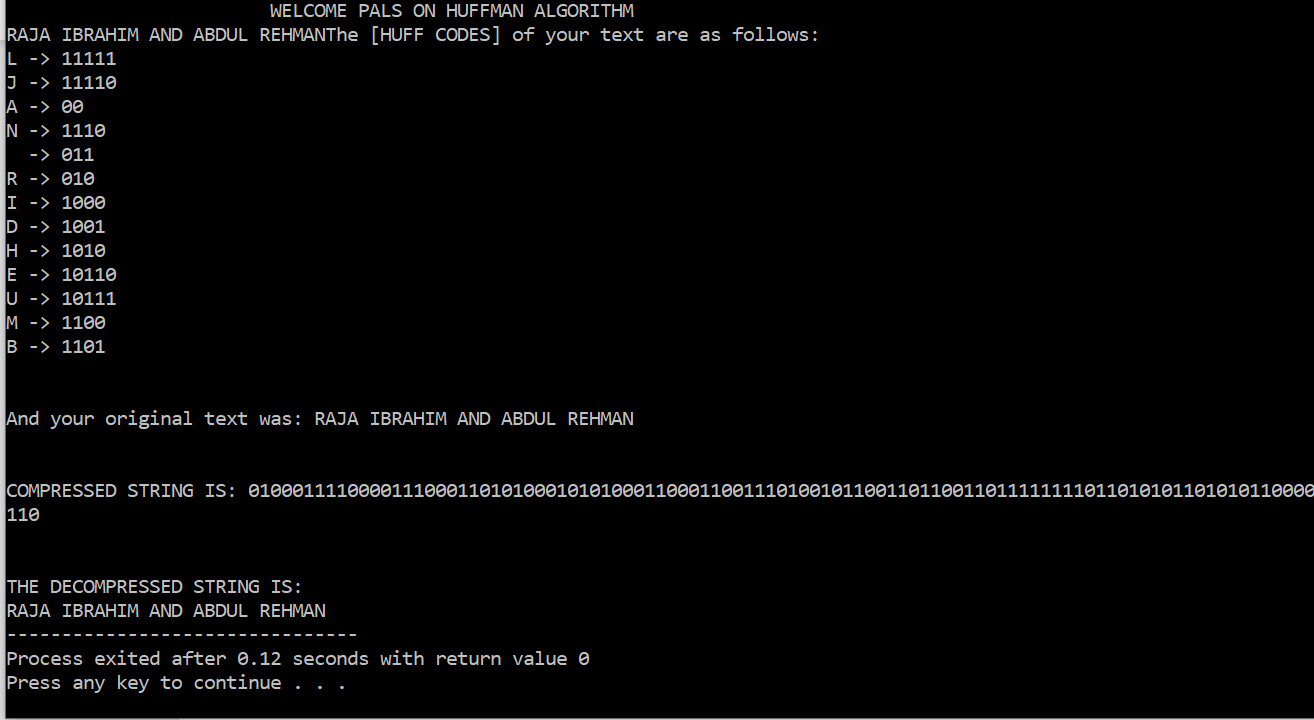


The file (input) will be read, and the text will be encoded, HUFF CODES will be displayed on the other file and decompressed string will be displayed on another file.

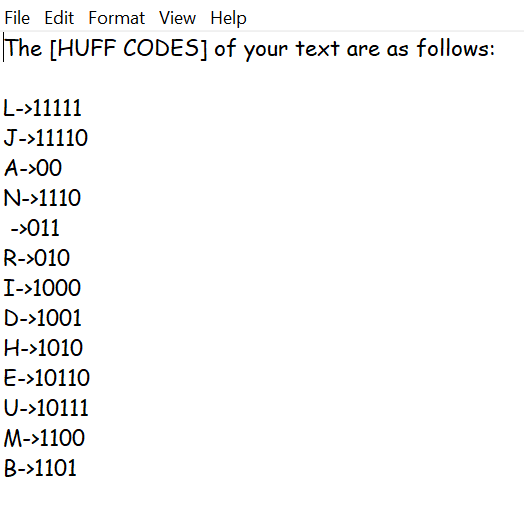
Input file:



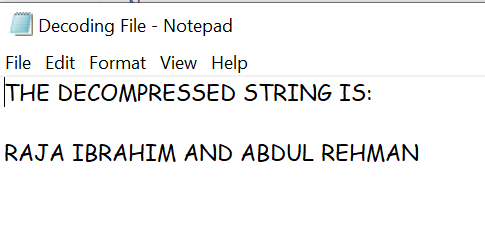
File will be read, console output is:



Encoded File:



Decoded file:



Output file:



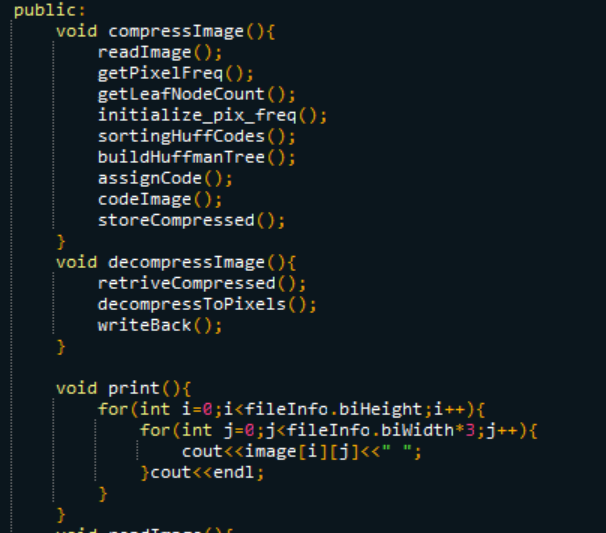
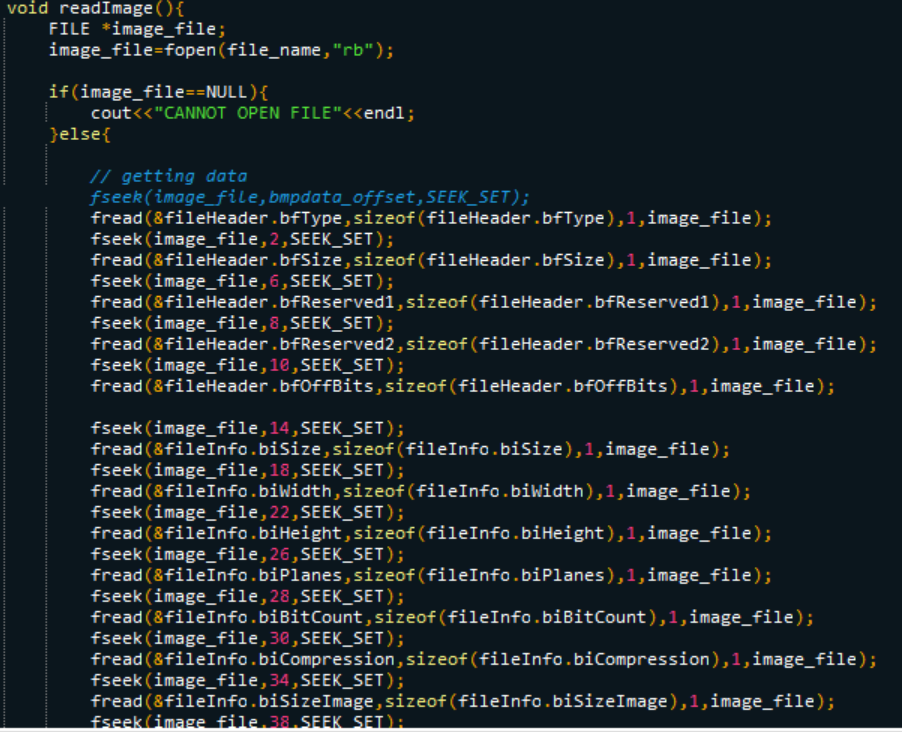
Images:

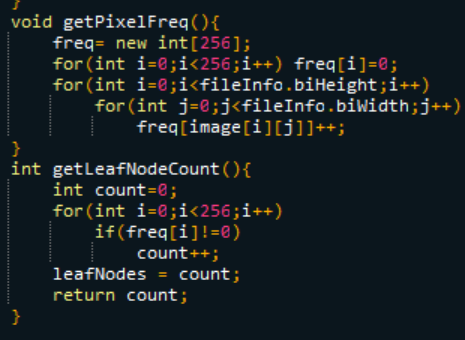
Image file is divided into three parts: header file, file information, image itself. While reading this file, we use offsets provided by standard BMP file format to read a particular data item. Once the data is read, the Huffman algorithm is applied to the pixel’s frequencies. Then both Huffman Codes and coded image are stored in different files. To decompress the image, we can simply read the stored Huffman Codes and the coded image and by applying Huff Codes to coded image we can get back our original image.

1. Classes:



1. Functions:

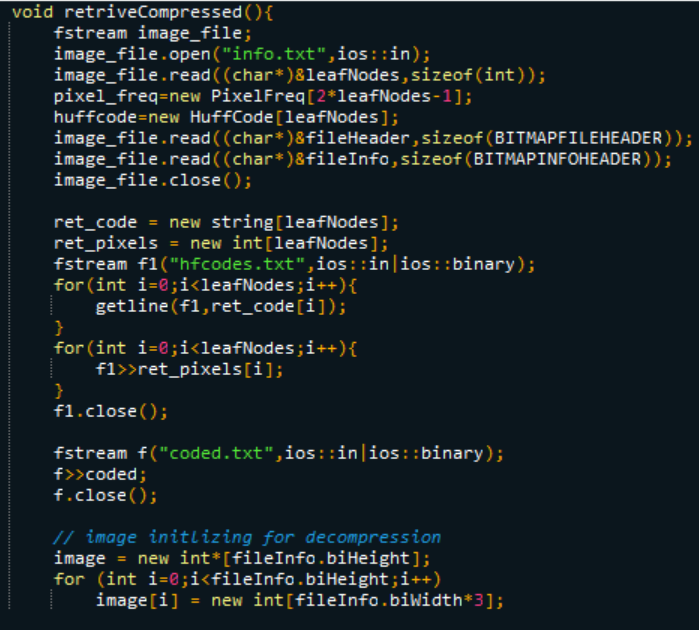
1. Building Huffman Tree:



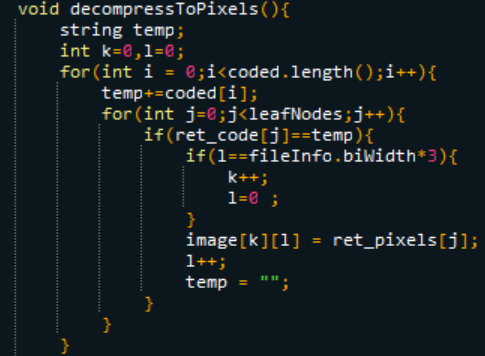
1. Storing our compressed image:



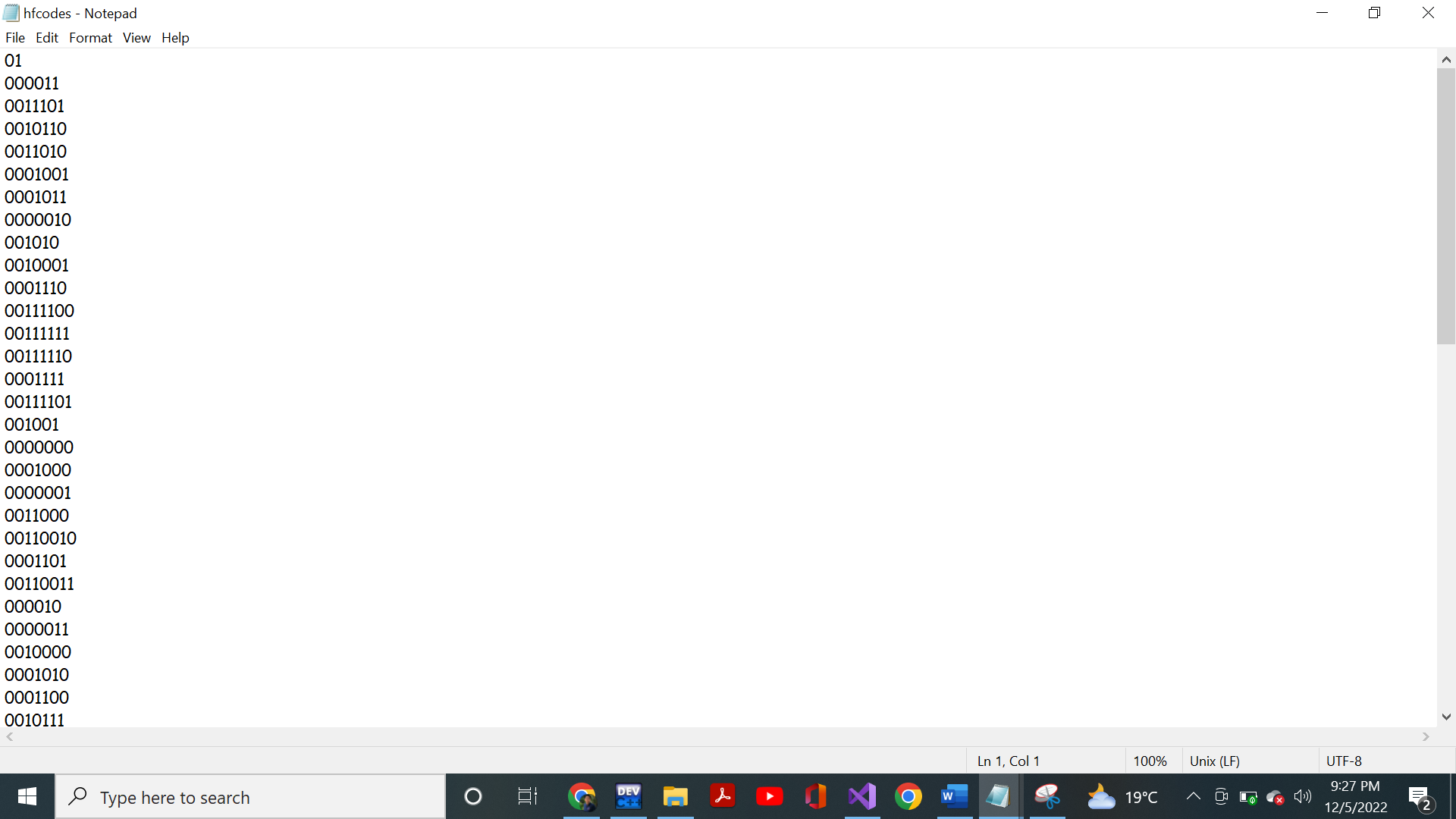
1. Retrieving compressed back:



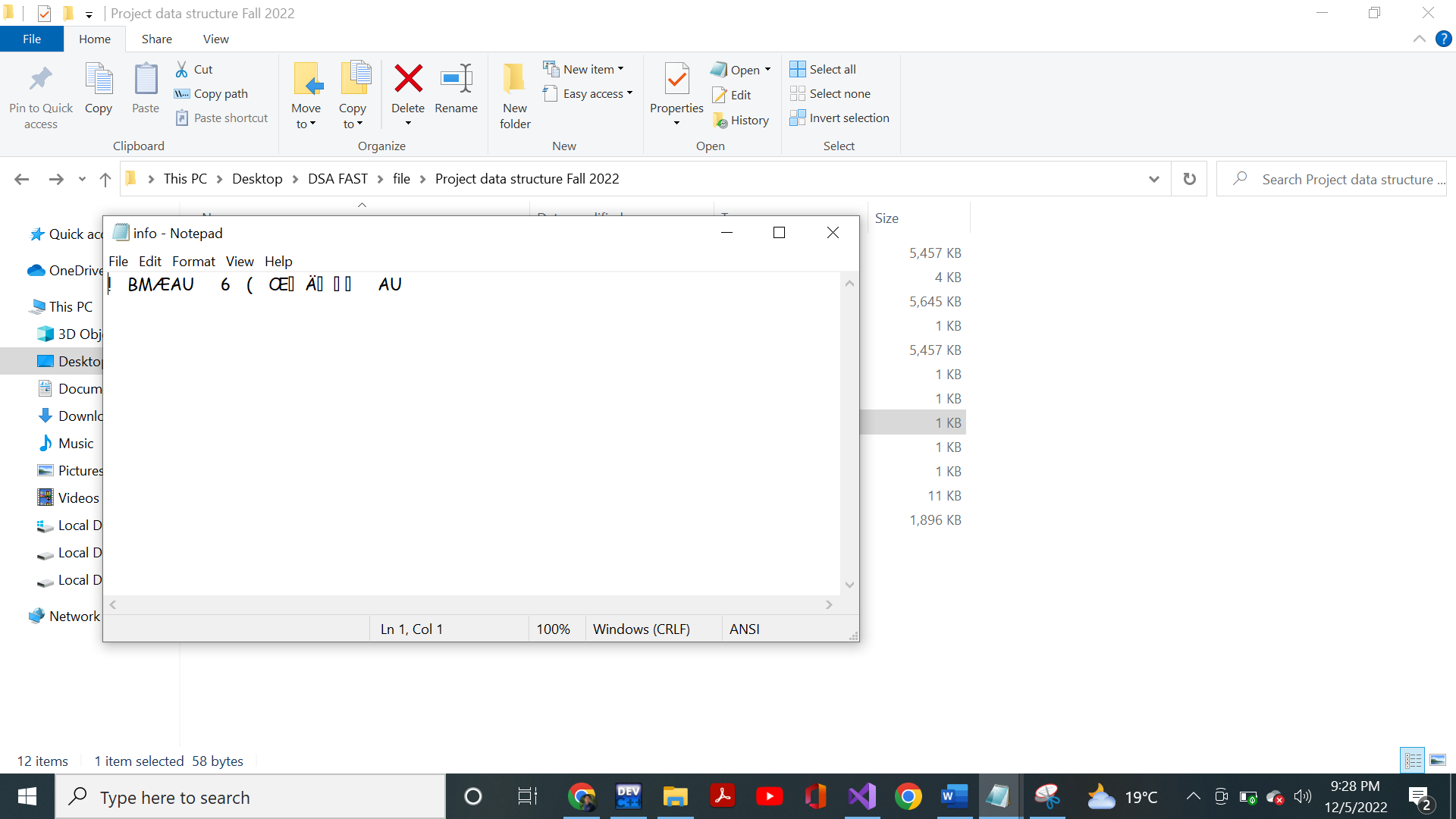
1. Decompress to Pixels:



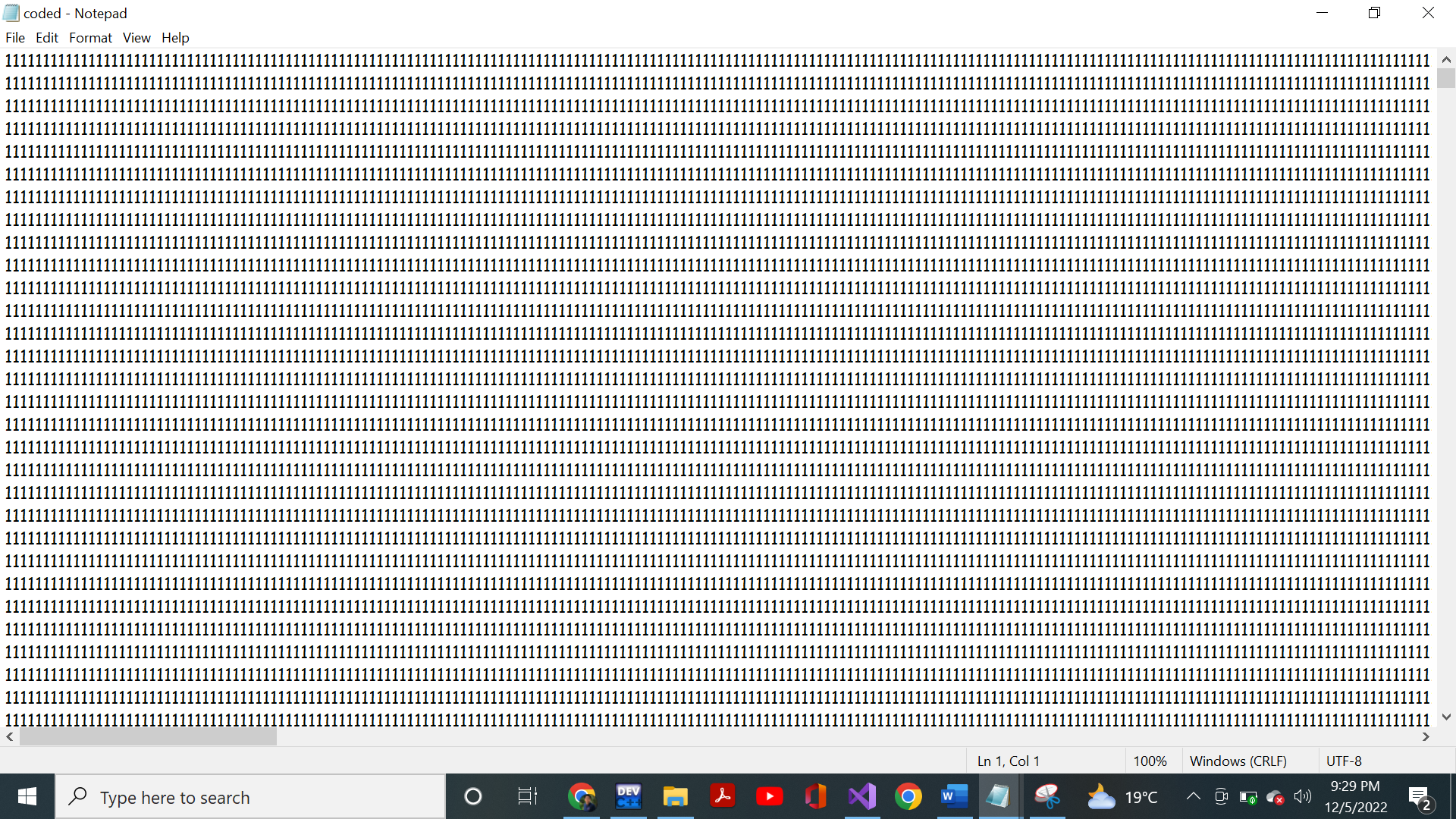
HF CODES:



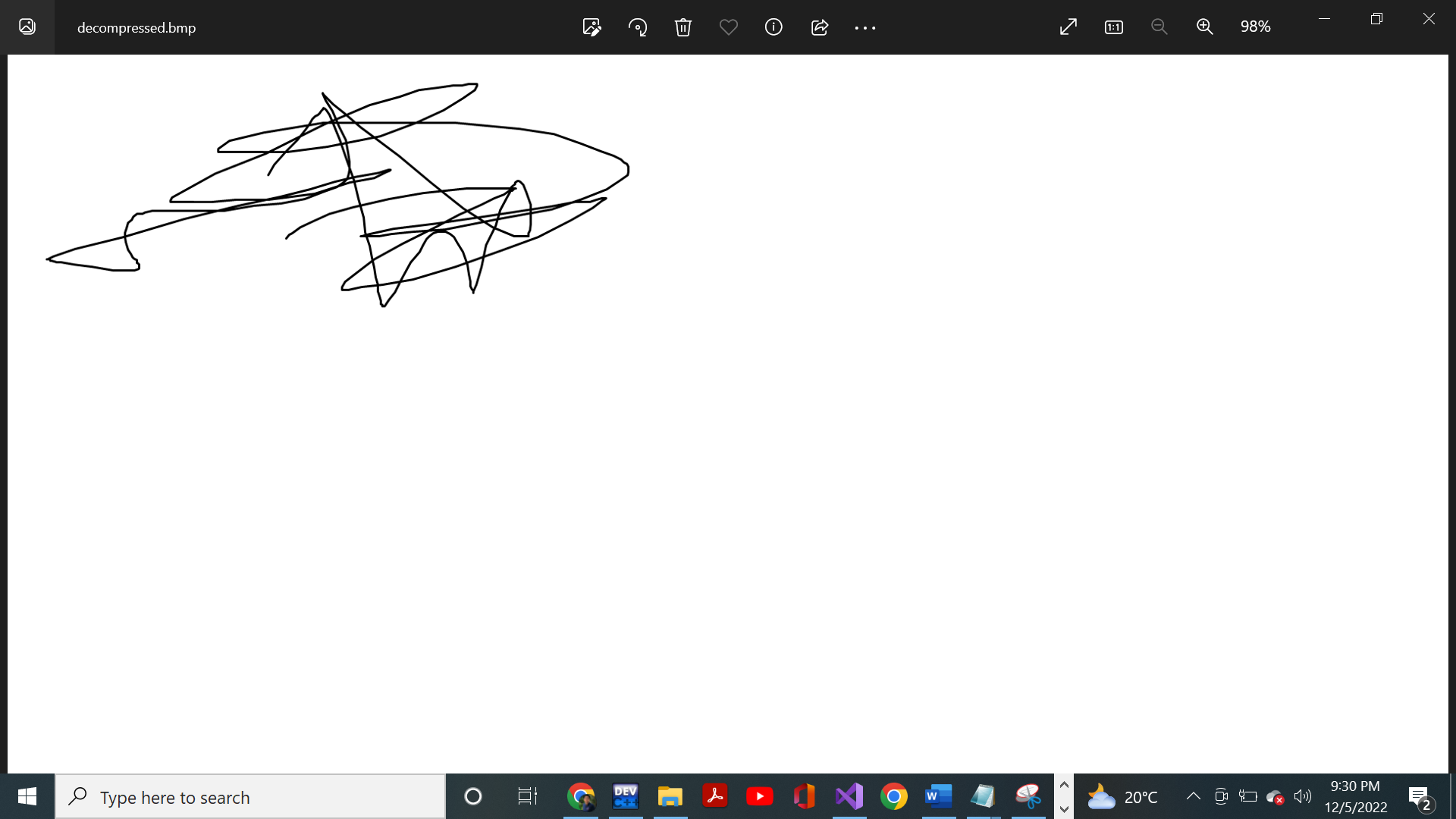
Header file:



Coded:



Decompressed:



Conclusion:

This was our project on Huffman Encoding Decoding. It enhanced our grasp on this course and improved our coding skills.

Thank You