ITCS 6114/8114: Algorithms and Data Structures
Programming Project 1: LZW Compression

NAME: SHARATH KANCHARLA

STUDENT ID: 801165873

LZW Compressor Program Design:

- In compressior.py the line **import argparse** imports the argparse module in order to pass bit length and filename as a command line argument.
- To specify the bit length, we use **args.bitlength** which reads the input passed as a command line argument.
- But the bit length can be < 9 or >16 and since the instructions were that the bit length should be > 9 and < 16 we need an **if** condition to check for bit length.
- When the condition is **False** the program prints the message and **sys.exit(0)** would exit without showing unnecessary errors.
- Else the Max Table Size will be set to 2^ (bit length).
- You need to pass the filename <input.txt> as a command line argument and to read the argument passed, we use **args.filename.**
- If an invalid file name or extension is passed the program prints message showing "Not valid" and terminates.
- I initialized table as an empty list and **chr(i)** returns ascii characters where **i** is in range (0, 256). We do this iteratively using **for** loop and append it to the table.
- The next few lines of code under 'Encoding Algorithm' performs tasks like encoding and updating table with new items, etc.
- The **while** loop iteratively reads symbols, looks for string+symbol in table and if it has found the code for string+symbol then it continues look by adding next symbol to string+symbol and so on.
- When there is no code available for string+symbol then the code for the string is appended to output and the table is updated with table.
- The **while** loop exhausts when there are no more symbols.
- The encoded output is printed by the program.
- Now the task is to create and write the 16-bit format of each encoded output to <input.lzw> file.
- For that I am using **code.to_bytes(2, byteorder = 'big')** where 2 is the no. of bytes for each code and byteorder is 'big endian'.
- An "input.lzw" file will be created in the LZW folder.

Data Structure Design:

- Program uses strings (Primitive Data Structure in Python) for content, string and symbol.
- Program uses lists (Non-primitive Data structure in Python) for table and output.

LZW Decompressor Program Design:

- In decompressor.py like in Compressor Program Design imports argparse module in order to pass bit length and filename as a command line argument.
- Like in Compressor Program Design the range of bit length should be valid or else the program terminates.
- The filename passed as a command line argument should be <input.lzw> or else the program terminates printing the error message.
- The formatted data from the file should be converted back to list of codes. I am iteratively appending **int.from_bytes(bytes_2[i:i+2], byteorder='big', signed=False)** to codes which converts 2 bytes of data at a time to its integer value.
- The codes serve as input to 'Decoding Algorithm'.
- I initialized table as an empty list and **chr(i)** returns ascii characters where **i** is in range (0, 256). We do this iteratively using **for** loop and append it to the table.
- The first code from the codes is decoded and stored in string.
- The **while** loop iteratively reads each code from codes, looks for code exits in table and if not exists string+string[0] (new_string) will be concatenated to content and table is updated with string+new_string[0] as a new item.
- Else table[code] is concatenated to content and table is updated with string+new_string[0] as a new item.
- The **while** loop exhausts when there are no more codes.
- The program prints the decoded output as a string.
- Now the task is to create and write the decoded output to <input_decoded.txt> file.
- I used **file.write(content)** to do that and an "input_decoded.txt" file will be created in the LZW folder.

Data Structure Design:

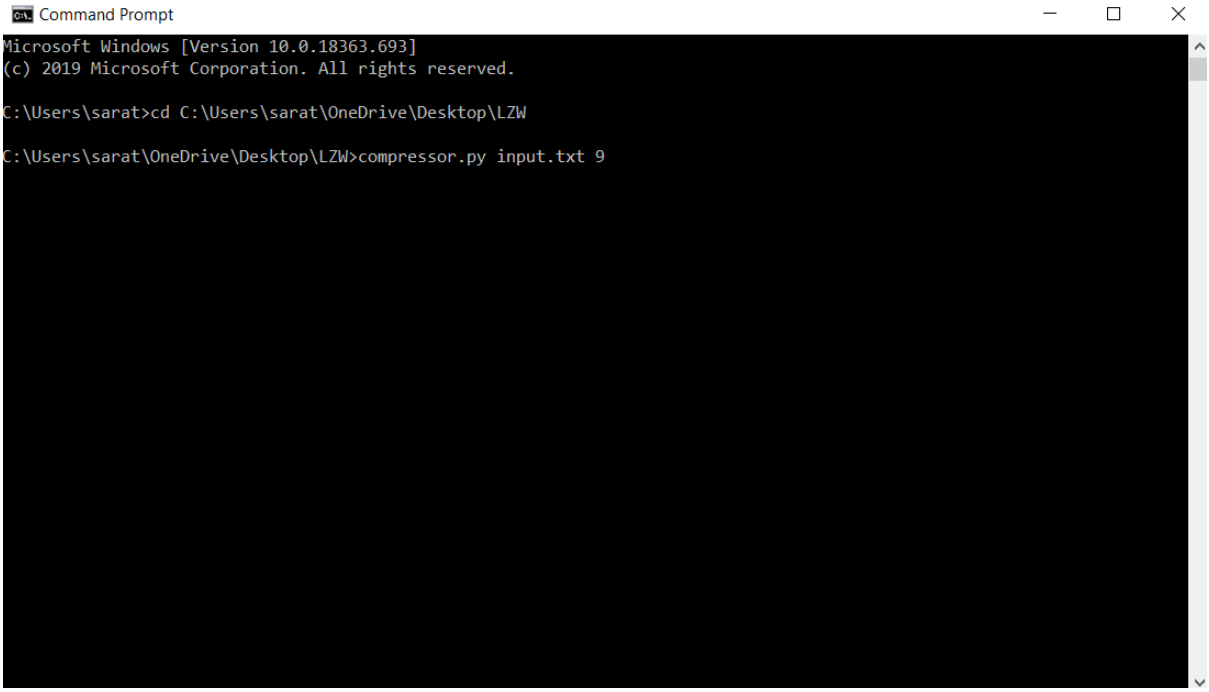
- Program uses strings (Primitive Data Structure in Python) for content, string, output and new_string.
- Program uses lists (Non-primitive Data structure in Python) for table and codes.

Summary:

- Both the programs work fine and gives the output required i.e. "input.txt" and "input_decoded.txt" have the same content even with multiple lines in the input file.
- The programs have been tested several times with results of both compressing and decompressing matching.
- There were no breakdowns while testing the programs.
- By default, the content stored in input.txt is "abbbab".

How to Run the Program:

- Python Version I am Using is Python 3.8.2. You can get it from https://www.python.org/downloads/release/python-382/. Under Files download the Windows x86-64 executable installer.
- Install and Set up python so that python programs run on cmd.
- Open Command Prompt.
- Change the directory to where the LZW folder is located.
- For running compression program enter **compressor.py input.txt 9**



- For running decompression program enter **decompressor.py input.lzw 9**