# CS 211 PA #4: Text Encoding / Compression

In this assignment, you will use hash tables and priority queues to perform compression on input files. The algorithm employed by this assignment is inspired by Huffman Coding Trees, which the book covers in Section 5.4. While it is not a requirement for you to read that section, you may get more out of the assignment if you do read its contents.

## Compression Algorithm

A popular method for compression is to perform value substation, replacing common strings with shorter strings, thereby saving space. Typically, this shorter sequence is represented using a binary number. Thus, the most common string in a file would receive the designation "0", the 2nd most popular "1", 3rd "10", 4th "11", 5th "001" and so on. This practice is the foundation of Huffman Coding Trees and the ubiquitous ZIP algorithm. Here's how to perform it on this assignment:

1. Record the frequency of each word in a given text file. Ignore empty lines and spaces.
2. Use the frequency recorded in step #1 to build a Max-Heap.
3. Removing items from the heap in order of importance, assign each word a unique "binary" string value.
4. Write to the output file the "binary" representation of the file.
5. Write the "binary" mapping to a separate file.

### Example

Using the algorithm above on the string "berries apples berries apples pears apples" (42 chars)

1. The frequencies of the words are berries: 2, apples: 3, pears: 1
2. The removal sequence from the heap would thus be: apples, berries, pears
3. The "binary" string assigned to each word would therefore be berries: "1", apples: "0", pears: "10".
4. The final compressed string would therefore be "1 0 1 0 10 0" (12 chars) for a compression ratio of 71%.

## Decompression Algorithm (optional / bonus)

For bonus points, you can try implementing the decompression algorithm, which is merely the inverse of the compression algorithm:

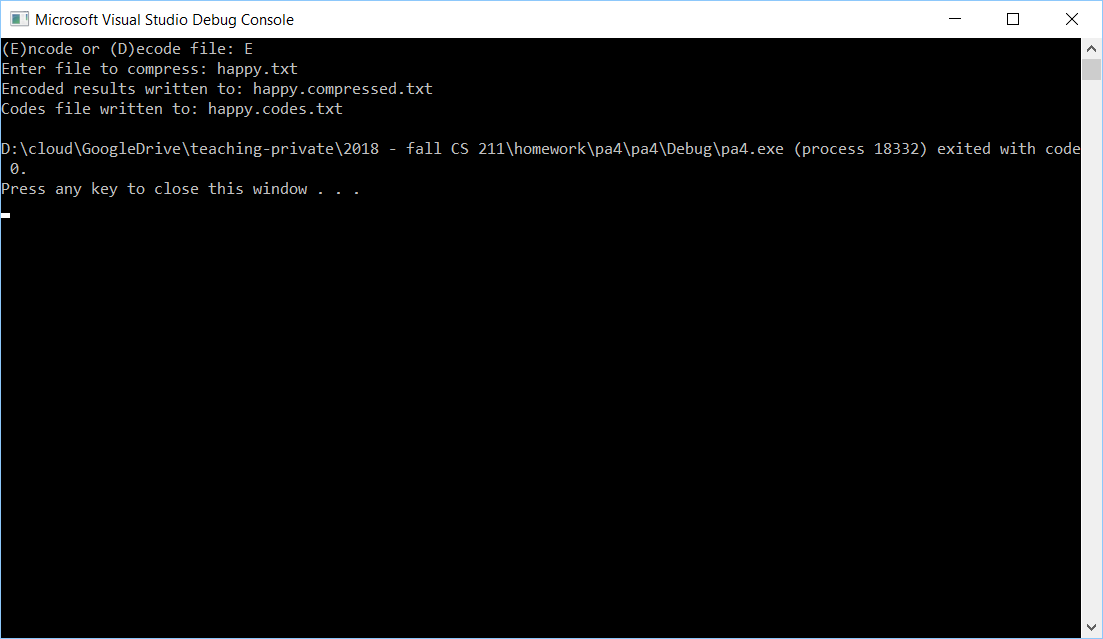
1. Load mapping file
   1. Load each mapped item (e.g. apples: 0) into a hash table.
2. Load compressed file
   1. For each string in the file, find the correct mapping (e.g. 0 -> apples), replace compressed version with uncompressed version

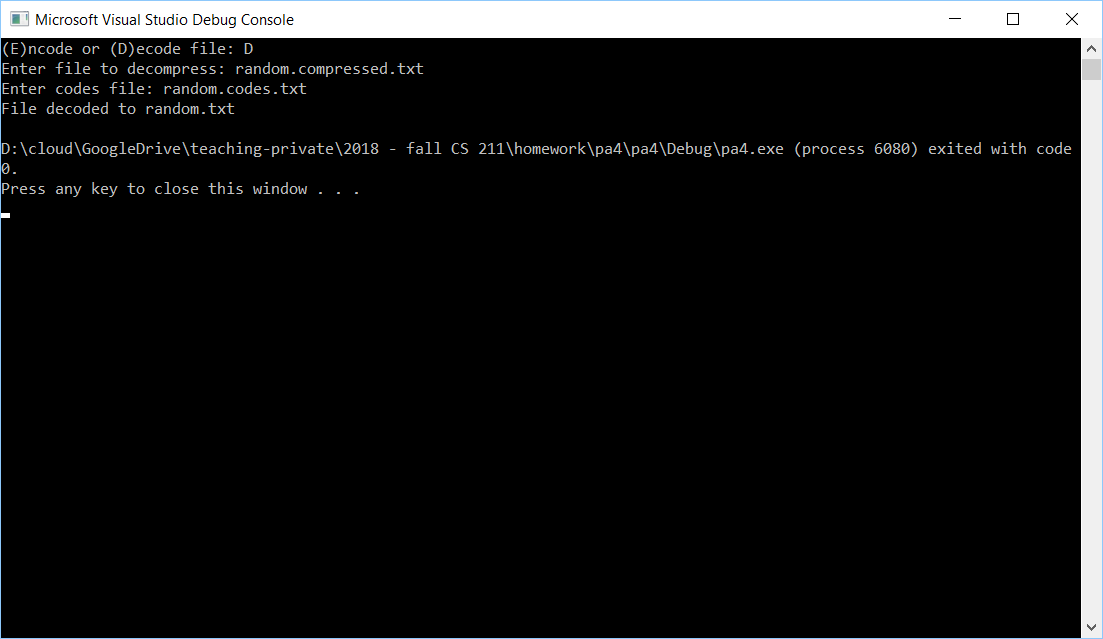
## File Formats

To maintain consistency across student submissions, please ensure that the encoded files have the following names: <source>.compressed.txt and <source>.codes.txt. Thus, if you load in "happy.txt", your program would generate the files "happy.compressed.txt" and "happy.codes.txt."

# Example Runs

Below, I provide you with two example runs, one for encoding and one for decoding.





# Testing Program

Provided with this file is a program that fully implements the compression scheme. I suggest using it to test your program against.

# Starter Code

There is no starter code provided for this project – you are free to implement the program however you see best. That being said, it might be useful for you to use the StringSplitter.h class included with this file.

# Possible Strategy for Getting Started

1. Note that you'll need a default animal whenever you start a fresh game. I suggest something common, e.g. "dog."
2. Don't worry about saving to the file. Get the tree constructed and the guessing game working before you think about saving and loading your tree.

# Header Comment, and Formatting

1. Be sure to modify the file header comment at the top of your script to indicate your name, student ID, completion time, and the names of any individuals that you collaborated with on the assignment.
2. Remember to follow the basic coding style guide. A basic list of rules is included with this document.

# Reflection Essay

In addition to the programming tasks listed above, your submission must include an essay that reflects on your experiences with this homework. This essay must be at least 350 words long. Note that the focus of this paper should be on your reflection, ***not*** on structure (e.g. introductory paragraph, conclusion, etc.). The essay is graded on content (i.e. it shows deep though) rather than syntax (e.g. spelling) and structure. Below are some prompts that can be used to get you thinking. Feel free to use these or to make up your own.

* Describe a particular struggle that you overcame when working on this programming assignment.
* Conversely, describe an issue with your assignment that you were unable to resolve.
* Provide advice to a future student on how he or she might succeed on this assignment.
* Describe the most fun aspect of the assignment.
* Describe the most challenging aspect of the assignment.
* Describe the most difficult aspect of the assignment to understand.
* Provide any suggestions for improving the assignment in the future.

## Deliverables

You must upload your program and reflection as a ZIP file through Canvas no later than midnight on Sunday, November 4, 2018.

## PA #4 Checkin

During lab on 10/29, you must demonstrate your current progress on your assignment.

## Grading Criteria

Your assignment will be judged by the following criteria (normalized to 100 points):

### Reflection essay (5pts)

* Your reflection meets the minimum requirements as specified earlier in this document.

### PA Checkin (10pts)

* You successfully demo your code during lab.

### Compressed File (60 pts)

* Your program creates an encoded file that matches my output.
* Your program creates a mapping file that matches my output.

### Decompressed File (BONUS: 20 pts)

* Your program can correctly decode a file.