**Objective:**

This assignment will introduce you to interprocess synchronization mechanisms in UNIX using named POSIX semaphores, pthread mutex semaphores, and pthread condition variables.

**Problem:**

For this assignment, you will modify your solution for [programming assignment 1](https://moodle2.cs.uh.edu/moodle/mod/vpl/view.php?id=1194) to comply with the restrictions explained below.

Given the information about the alphabet and the compressed file as input from STDIN, you need to implement a multithreaded Huffman decompressor based on the following steps:

* + Read the input from STDIN (the Moodle server will implement input redirection to send the information from a file to STDIN). The input has the following format:

4  
A 3  
C 3  
B 1  
D 2  
11 1 3 5  
0 0 2 4  
101 6 8  
100 7

* + - * The first line has a single integer value representing the number of symbols in the alphabet (n).
      * The next n lines present the information about the alphabet. Each line presents the information about a symbol from the alphabet:

A character representing the symbol.

An integer representing the frequency of the symbol.

* + - * The final n lines present the information about the compressed file. Each line presents the information about a compressed symbol:

A string representing the binary code of the symbol.

A list of m integers (where m is the frequency of the symbol) representing the positions where the symbol appears in the message.

* + Generate the Huffman Tree based on the input.
  + Create n POSIX threads (where n is the number of symbols in the alphabet). Each child thread executes the following tasks:
    - * Receives the Huffman tree and the information about the symbol to decompress (binary code and list of positions) from the main thread.
      * Uses the Huffman tree to determine the character from the original message based on the binary code.
      * Stores the decompressed character (as many times as needed based on the list of positions) on a memory location accessible by the main thread.
      * Print the information about the assigned symbol using the output message from the example below.
  + Print the original message.

Given the previous input, the expected output is:

Symbol: A, Frequency: 3, Code: 11  
Symbol: C, Frequency: 3, Code: 0  
Symbol: D, Frequency: 2, Code: 101  
Symbol: B, Frequency: 1, Code: 100  
Original message: CACACADBD

**NOTES:**

* + **You can safely assume that the input files will always be in the proper format.**
  + **You cannot use global variables. A 100% penalty will be applied to submissions using global variables.**
  + **You must define the critical sections following the guidelines that we discussed in class.**
  + **You must use POSIX threads. A penalty of 100% will be applied to submissions using a thread library other than the pthread library.**
  + **You can only use named POSIX semaphores, pthreads mutex semaphores, or pthreads condition variables to achieve synchronization. Using pthread\_join or sleep to synchronize your threads is not allowed (you must use pthread\_join to guarantee that the parent thread waits for all its child threads to end before ending its execution). A penalty of 100% will be applied to submissions using the previous system calls to synchronize the child threads.**
  + **You cannot use different memory addresses to pass the information from the parent thread to the child threads.**
  + **You must use the output statement format based on the example above.**

Requested files

* main.cpp
* huffmanTree.h
* huffmanTree.cpp