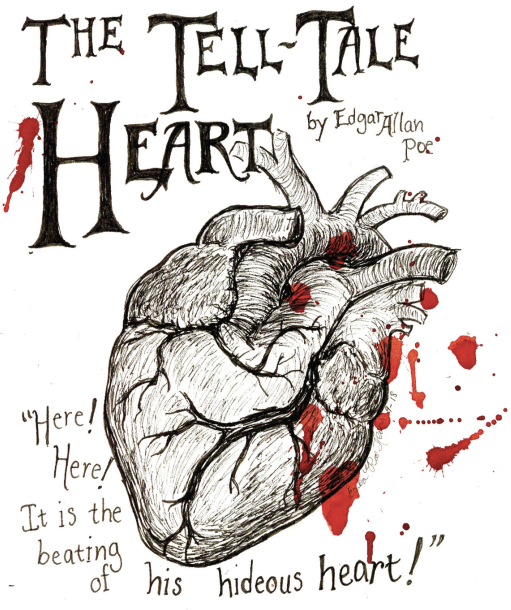
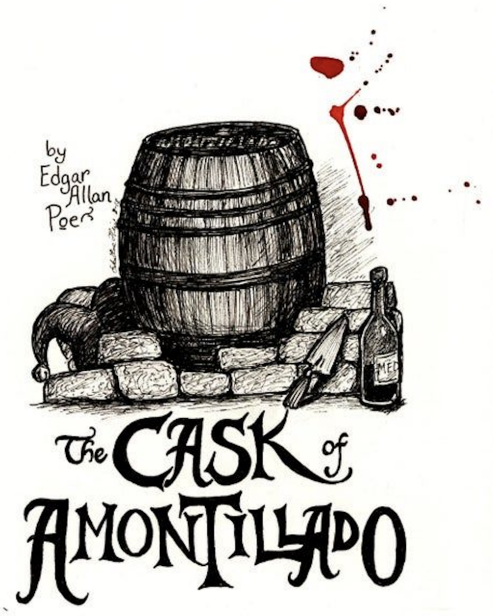
PROGRAMMING ASSIGNMENT #5
CS 2223 B-TERM 2024
CLOSED HASHING
&
DIJKSTRA'S ALGORITHM

ONE HUNDRED POINTS
DUE: THURSDAY, DECEMBER 5, 2024 11 PM





"Those who are mad know nothing. . . "

In pace requiescat!

1. (10 Points) Read a file and hash the words:
   Read a text file (there are several named in the table below and posted on Canvas) and hash the words using the suggested hash function from *Levitin*, page 269:

$$h \leftarrow 0; \textbf{ for } i \leftarrow 0 \textbf{ to } s-1 \textbf{ do } h \leftarrow (h * C + ord(c_i)) \mod m,$$

   where:

   h is the computed hash value,
   s is the length of the word being hashed,
   c_i is the i^{th} character of the word,
   $ord(c)$ is the numerical value of character c in the alphabet in use,
   C is a constant larger than every $ord(c_i)$, and
   m is the modulus defining the size of our hash table.

   We will take $ord(c)$ to be the ASCII value of the characters in each word. Thus, $ord(c) \in \{39, 45, 65, 66, \ldots, 90, 97, 98, \ldots 122\}$ for $c \in \{', \text{-}, A, B, \ldots, Z, a, b, \ldots, z\}$, respectively. All other characters are discarded; we define words as unbroken sequences (strings!) of consecutive alphabetic characters, both upper and lower case, plus apostrophes and hyphens[1], in the lines read for the file. (Java uses Unicode, but ASCII is still under there somewhere—you just need to dig it out.)

   We'll start simply with "TheRavenB24.txt" text file, and we will take $C = 123$ and $m = 1000$, even though it's not prime. These represent parameters that can be changed to alter the shape and composition of our resulting hash table. We'll increase the size of our table when we use the appended files.

   You are free to experiment with changing these parameters on your own, of course!

File	Table Size	Grading Comment	Extraneous Comment
TheRavenB24.txt	1000	Test Version	'answers' to be supplied on weekend
TellTaleHeartB24.txt	790	To Be Graded	
CaskOfAmontilladoB24.txt	991	To Be Graded	Something *excellent* Happens
Heart prepended to Cask	1499	To Be Graded	Where are the poet's three names?
Cask prepended to Heart	1499	To Be Graded	Where are the poet's names now?

[1]Thus, *blood-spot* ("The Tell-Tale Heart") and *d'or* ("The Cask of Amontillado") are each individual words.

2. (20 Points) Create a Hash Table using Closed Hashing (Open Addressing):

   a. Build a hash table of size n, i.e. entries from 0 to $n-1$, from the hash values computed in Part 1 above. You should have the user enter the size of the table to be created, i. e. n. Load the words into the table in the order they occur in the file, *discarding duplicates*. Some of our tables might get close to full, but you needn't worry about re-sizing them—we will examine the effects of their load factors in Part 3 below.

   b. Display the table, starting with table entry 0, in lines of the form:

      Hash Address, Hashed Word, Hash Value of Word

      (Remember, the Hash Address will not match the Hash Value when the resolution of a collision forces a word to be cascaded down the table. Also, remember that the table should be thought of as a circular list so that a word which cascades past the bottom end of the table gets wrapped to the top in looking for an open place in the table.)

   Note: Your program may perform Parts 1 & 2 simultaneously, if you wish.

3. (25 Points) Explore the Hash Table:

   Add to your code routines that answer these questions. Some answers may not be unique. You may resolve these issues by presenting either *any* correct answer or *all* correct answers.

   a. How many non-empty addresses are there in the table? What does that make the load factor, α, for our table?

   b. What is the longest empty area in the table, and where is it?

   c. What is the longest (largest) cluster in the table, and where is it?
   Note: It might wrap from the end of the table back to the beginning.

   d. What hash value results from the greatest number of distinct words, and how many words have that hash value?

   e. What word is placed in the table farthest from its actual hash value, and how far away is it from its actual hash value?

4. (5 Points) Explore the effects of order and more. . . :

   Create two new files by appending the contents of "The Tell-Tale Heart" and "The Cask of Amontillado", and build hash tables of size 1499 from these new files as before. Do our poet's three names (included atop the file of "The Tell-Tale Heart") land in the same places in both tables? Why? Why not?

   Can you determine how many words the files have in common?

   Make sure your code prompts the user for the desired input so that your grader will know what to do.

   You can assume that the table size will be large enough for the given file—that is, you need not check to see whether the combination of file and table size will result in an infinite loop.

   You also need not check that the file exists, but this would be good practice for handling *file-not-found* exceptions.

   I'll provide sample answers from a couple of different text files and different sizes of tables for you to check against as the due date draws closer.

"*All* that we see or seem is but a dream within a dream."

5. (40 Points) Implement Dijkstra's Algorithm with weighted graphs

You can hardcode this matrix–we have a lot to get done in a short time.–but you can also read it from a file if you prefer. Your code should ask for input from the console for start and destination nodes and use Dijkstra's algorithm to find the "length" of the shortest path between them. It should also display the sequence of nodes that constitute this shortest path.

Alternatively, you can merely ask for a source vertex and produce the shortest path to every vertex (in true Dijkstra form).

0	53	10	12	0	0	0	0	0	0
53	0	33	0	2	0	101	0	0	0
10	33	0	9	30	18	0	0	0	0
12	0	9	0	0	17	0	0	6	0
0	2	30	0	0	14	123	122	0	0
0	0	18	17	14	0	0	137	7	0
0	101	0	0	123	0	0	8	0	71
0	0	0	0	122	137	8	0	145	66
0	0	0	6	0	7	0	145	0	212
0	0	0	0	0	0	71	66	212	0