Assignmient 7

Author Identification

DESIGN.pdf

Reuben T. Chavez

March 6, 2022

Pseudeocode

```
# Psuedocode for identify.c

function usage is
    input: executable
    output: void


function main is
    input : argument count argc and argument vector argv
    output: zero to exit program

    - Perform getop operators to determine:

    -
```

```
#Pseudocode for Nodes

##Libraries

import stdint.h

define type Node;

Initialize struct Node with:
    - char pointer to word
    - unsigened inter of 32 bit to count

Function Node Create:
    input: char pointer
    output: a pointer to a Node type

    Allocate space for Node n with a size of Node
    Allocate space for word in Node
    Set Node's word to copy of input
    return Node n
```

```
Function Node Delete:
    Input: Double pointer n
    Output: None since functioin is void

    - free word since space was allocated for it
    - free contents in n
    - set to null

Function print node:
    input: pointer to node
    output: Nothing function is void

    print the data item withn the pointer node
```

```
#Pseudocode for pq.c

#Libraries
import node
import stdbool
import stdint


struct PriorityQueue
    - contains head
    - contains tail
    - contains capacity
    - contains Node array

Function Create Priority Queue:
    Input: An unsigned intger of 32 bits
    Output: Priorit Queue pointer

    - Allocate space for a Priority Queue pointer
    - Initialize head, tail, capacity, and the Node array if the pq
        is not NULL

Function Insertion Sort:
    Input: A Priority Queue and Node
    Output: Nothing function is void
    - For iteration of Priorty queue:
        - set j to current index
        - create temp of current index in PQ array
        - While j is greater than 0 and the temp is greater than the
            last index
```

```
            - set array at index j to the last insex
            - subtract j by 1
        - set the Priorty Queue on index j to temp

Function pq delete:
    Input : Double pointer to Priority Queue
    Output : Nothing function is void

    - If the input is not null, free double pointer and set
        previous node to NULL

Function pq full:
    Input : Double pointer to Priority Queue
    Output: boolean

    - rturn that given pq is either full or not


Function pq empty:
    Input : Double pointer to Priority Queue
    Output: boolean

    - rturn that given pq is either empty or not


Function pq size:
    Input : Priorty Queue pointer
    Output: Unsigned 32 interger

    - return the the top node in pq


Function enqueue:
    Input: Priorty Queue pointer and Node pointer
    Output: boolean

    - if Priorty Queue not null,
        - if empty return false
        - add node to head of pq
        - Resort the tail node to the in the pq using a sorting
            algorthim
    -return true to signify that the pq was succefully enqued

Function dequeue:
    Input: Priorty Queue double pointer and Node pointer
```

```
    Output: boolean

    - if Priorty Queue not null,
        - if full return false
        - remove node from tail of pq
        - Resort the tail node to the in the pq using a sorting
            algorthim
        - sub top by 1
    -return true to signify that the pq was succefully enqued

Function pq print:
    Input: Priorty Queue Node
    Output: Nothing the function is void

    - Print all items with pq
```

```
#Pseudocode for stack.c

#Libraries
import node
import stdbool
import stdint
import stdlib

Stack struct:
    - contains top
    - contains capacity
    - contains double pointer node array

Function stack create
    Input: unsigned 32 bit integer
    Output: Pointer to Stack

    - Allocate memory fro STACK object
    - Initilize items within Stack struct
    - return stack pointer

Function stack delete:
    Input: Stack pointer
    Output: Nothing function is void

    - Delete specifed stack, and set previous stack to NULL

Function stack empty:
```

```
    Input : Stack pointer
    Output: boolean

    - return if the stack is empty

Function stack full:
    Input : Stack pointer
    Output: boolean

    - return if the stack is full

Function stack push:
    Input: Stack pointer and pointer to node
    Output: boolean

    - Check if the stack is not full, if its add more space to stack
    - Add stack top and set node pointer equal to


Function stack pop:
    Input: Stack pointer and double pointer to node
    Output: boolean
```

```
#Pseudocode for code.c
```

```
#Pseudocode for huffman.c
```

```
#Pseudocode for
```
