**Approach to PROJ 4, LRU Buffer Pool**

Riley Dorough / Chandler Whitley

26MAR20

This project was completed through the collaborative efforts of the genius Chandler Whitley & the innovative Riley Dorough.

**Least Recently Edited Buffer Pool**

**Count\_SOL.h**

Creates a llist object pointer of type <E> and a

Count\_SOL variables:

* Private access to llist object pointer to store items
* Integer to track compares

Count\_SOL class (construct & destruct):

* Constructor creates a new llist object of type <E>

Count\_COL methods:

* Find() checks to see if requested value is in list if it’s not, it will add the value to the list by calling the add method, then should call reorder to reformat the list
* Add() creates a link to the new value and appends it to the list
* getCompares() returns the number of comparisons
* size() returns the size of the list
* printlist() outputs the full length list
* printlist( int ) outputs the list starting from a given integer
* reorder() Sorts the list based on frequency of elements accessed.

**MTF\_SOL.h**

Creates a llist object pointer of type <E> and a

MTF\_SOL variables:

* Private access to llist object pointer to store items
* Integer to track compares

MTF\_SOL class (construct & destruct):

* Constructor creates a new llist object of type <E>

MTF\_COL methods:

* Find() checks to see if requested value is in list if it’s not, it will add the value to the list by calling the add method, then should call reorder to reformat the list
* Add() creates a link to the new value and appends it to the list
* getCompares() returns the number of comparisons
* size() returns the size of the list
* printlist() outputs the full length list
* printlist( int ) outputs the list starting from a given integer
* reorder() Sorts the list based on frequency of elements accessed.

**Transpose\_SOL.h**

Creates a llist object pointer of type <E> and a

TSOL variables:

* Private access to llist object pointer to store items
* Integer to track compares

TSOL class (construct & destruct):

* Constructor creates a new llist object of type <E>

TSOL methods:

* Find() checks to see if requested value is in list if it’s not, it will add the value to the list by calling the add method, then should call reorder to reformat the list
* Add() creates a link to the new value and appends it to the list
* getCompares() returns the number of comparisons
* size() returns the size of the list
* printlist() outputs the full length list
* printlist( int ) outputs the list starting from a given integer
* reorder() Sorts the list based on frequency of elements accessed.

**Main.cpp**

1. Add the required values using the count sorted order list add function
2. Find the required values using the sorted order list find function.
3. Repeat step one and two using the move to front and transpose methods.